Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Brožič

Optično branje univerzalnega plačilnega naloga

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Borut Batagelj

Ljubljana 2013

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: 00401/2013
Datum: 03.04.2013

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

Kandidat: **MATEJ BROŽIČ**

Naslov: **OPTIČNO BRANJE UNIVERZALNEGA PLAČILNEGA NALOGA**
THE UNIVERSAL PAYMENT ORDER OPTICAL READING

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Ročno vnašanje besedila iz univerzalnega plačilnega naloga je zamudno početje. Zaradi tega se je v bankah pojavila potreba po avtomatizaciji tega postopka. V ta namen so bili razviti posebni čitalniki, katere najdemo nameščene na različnih krajih od bank do trgovskih centrov.

Kandidat naj pregleda možnost avtomatizacije zajema vsebine univerzalnega plačilnega naloga s pomočjo metod računalniškega vida. Izdela naj sistem optičnega branja plačilnega naloga in na ta način omogoči tudi uporabnikom, ki nimajo čitalnikov, avtomatsko vnašanja plačilnega naloga v sistem zgolj s pomočjo zajete slike.

Mentor:

viš. pred. dr. Borut Batagelj

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matej Brožič, z vpisno številko **63070250**, sem avtor diplomskega dela z naslovom:

Optično branje univerzalnega plačilnega naloga

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Boruta Batagelja,
- 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 25. junija 2013 Podpis avtorja:

Zahvaljujem se viš. pred. dr. Borutu Batagelju, za vso strokovno pomoč, nasvete, mentorstvo in vodenje pri izdelavi in pisanju diplomske naloge.

Hvala staršem, bratu in Sabini, ki so mi pomagali skozi študij in me spodbujali vsa leta.

Zahvaljujem se tudi Luki Godniču, ki mi je pomagal in svetoval pri izdelavi programskega dela diplomske naloge in ostalim sodelavcem za pomoč pri testiranju.

Kazalo

Slike

Tabele

Povzetek

Dandanes je plačevanje z univerzalnim plačilnim nalogom zelo pogosto. Z njim se lahko plačuje v bankah, na poštah, na nekaterih bencinskih servisih in tudi na blagajnah nekaterih trgovin. Uporabniki spletnih in mobilnih bank lahko plačujejo tudi od doma preko spleta ali pa s pametnim telefonom. Za izvršitev plačila univerzalnega plačilnega naloga je potrebno iz njega prebrati podatke in jih vnesti v obrazec oziroma program za delo s plačilnimi nalogi. Uporabniki spletnih in mobilnih bank in tudi zaposleni v bankah še danes to počnejo ročno, tako da podatke pretipkajo v program (razen ponekod, kjer imajo na voljo čitalnike). To je seveda kar zamudno opravilo. Glavni cilj bank je, da komitentom čim bolj olajšajo delo, zato smo se v sodelovanju z Banko Koper odločili razviti sistem, ki bi omogočal enostavno plačevanje položnice s slikanjem le-te, sistem pa bi nato s slike prebral podatke in jih posredoval programu za delo z univerzalnimi plačilnimi nalogi. Ta sistem je obravnavan v diplomskem delu.

V uvodu diplomskega dela predstavimo univerzalni plačilni nalog in pregledamo že obstoječe podobne sisteme ter nekatere na kratko predstavimo. Sledi predstavitev razvoja sistema in tehnologij ter metod, uporabljenih pri razvoju. Tukaj je predstavljen način in postopek predprocesiranja slike, s katerim najdemo polja, v katerih so podatki, ki nas zanimajo. Nato pa je predstavljeno še branje podatkov iz najdenih polj. Na koncu se delo posveti še predstavitvi rezultatov, dobljenih z opisanim sistemom, ter nadaljnjim planom dela in izboljšav.

Ključne besede:
Univerzalni plačilni nalog, optično razpoznavanje znakov, predprocesiranje slike, računalniški vid, detekcija robov

Abstract

Paying bills with the universal payment order is very common in these days. It is used to pay bills in banks, post offices or at home, with the use of web banks and mobile banks on smart phones. To use the universal payment order in payment, it is necessary to read the data from it and enter the data in program for work with orders. The users of web and mobile banks and bank employees mostly do the data entering manually, by typing every character with their hands, which can be quite time-consuming task. Because the main goal of every bank is to simplify things for its costumers, we decided in cooperation with Banka Koper, to develop a system which would enable simple use of payment order. The system would be able to read the universal payment order data from its image and fill in the program for work with orders, so there would be no need of manually filing the data. This system is presented in this thesis.

In the first part of the thesis we present the universal payment order and look trough some of the existing similar systems and present some of them. In the second part the development of the system and used technologies and methods are presented. In this part, the preprocessing of the image, used to find the fields we are interested in is presented. The optical character recognition of the data inside these fields is also presented in this part. The last part of the thesis looks over results that the system gives us and the future planned work and improvements.

TABELE

Key words:
universal payment order, optical character recognition, image preprocessing, computer vision, edge detection

Poglavje 1

Uvod

V diplomski nalogi so predstavljena različne programske obdelave slike in optično razpoznavanje znakov v sliki, ki so nato uporabljeni v sistemu za optično razpoznavo in branje slike univerzalnega plačilnega naloga. Prva vprašanja, ki smo si ji zadali, so bila, kaj sploh je univerzalni plačilni nalog, kakšen je njegov tehnični standard in ali obstaja že kakšna rešitev za branje le-tega ali kakšnega drugega obrazca nasploh.

Uporaba univerzalnega plačilnega naloga se je začela s pričetkom novembra leta 2010. Takrat so banke pričele uvajati ta nov obrazec in iz uporabe umikati obstoječe obrazce. V okviru teh prilagoditev so banke sprejele tudi tehnični standard novega univerzalnega plačilnega naloga [1].

Uporablja se za naslednje storitve [2]:

- negotovinska plačila,
- gotovinska plačila,
- pologe gotovine,
- dvige gotovine.

V tehnični dokumentaciji univerzalnega plačilnega naloga [3] je določeno, da se obrazec izpolnjuje na dva načina.

- Ročno izpolnjevanje, z velikimi tiskanimi črkami, kjer se v posamezna okenca v poljih vpiše ena črka oziroma številka. Obvezna je uporaba pisal s temno barvo (črna, modra), vsebina vseh polj pa mora biti poravnana levo, razen znesek, ki je poravnan desno.

- Strojno izpolnjevanje, s predpisano neproporcionalno pisavo Courier New 12 CPI, skladno s tehničnim standardom. Pri strojnem izpolnjevanju, črk oziroma številk ni potrebno vpisovati točno v okenca, biti morajo le v pravem polju. Znesek in datum sta formatirana.

V diplomski nalogi bomo obravnavali le strojno izpolnjene univerzalne plačilne naloge.

Naprej je v tehnični dokumentaciji določeno, da je univerzalni plačilni nalog širok 210 milimetrov in visok 101.6 milimetrov, po širini pa je razdeljen na talon, širine 60 milimetrov in nalog za plačilo, širine 150 milimetrov. Talon je namenjen stranki in nas njegova vsebina ne zanima, zato smo se poglobili le v nalog za plačilo. Slika 1.1 je slika celotnega univerzalnega plačilnega naloga, z označenim nalogom za plačilo in vsemi polji, ki nas zanimajo. Nalog za plačilo je po višini razdeljen na 3 vodoravne predele.

- Zgornji predel, ki vsebuje polja o plačniku in je visok 39 milimetrov. V tem delu nas izmed vseh polj zanimata le polji koda namena in namen plačila.

- Srednji predel, ki vsebuje polja o prejemniku in je visok 37.5 milimetrov. V tem predelu se nahajajo polja:

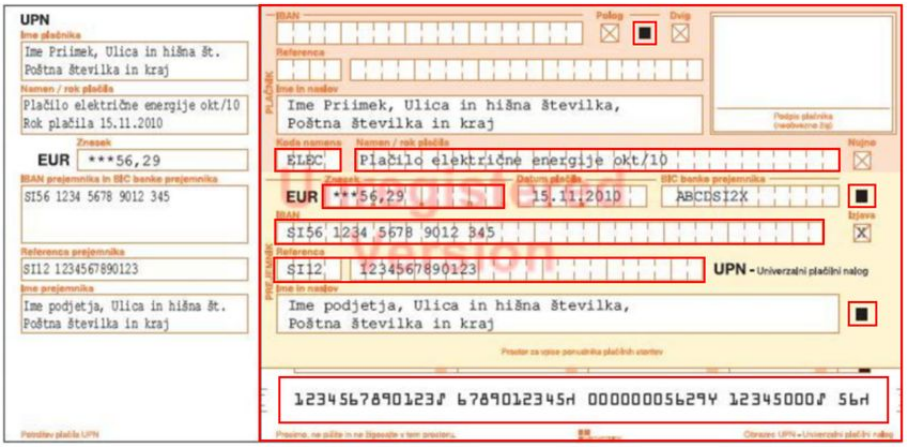
  - znesek,
  - datum plačila,
  - bic banke prejemnika,
  - iban,

- izjava,
- referenca,
- ime in naslov.

Izmed teh polj nas za obdelavo zanimajo znesek, iban in referenca.

- Spodnji predel, imenovan tudi vrstica OCR, ki je namenjen optičnem zapisu podatkov. Vrstica OCR je določena v skladu s standardi iso, znaki pa morajo biti tiskani v črni barvi in pisavi OCR-A1 EURO-BANKING.

Ta del smo v našem sistemu uporabili kot preverbo za testiranje prebranih podatkov, saj v njem najdemo večino podatkov, zapisanih v srednjem delu. Na nalogu za plačilo opazimo tudi 3 črne kvadratke, ki so namenjeni ravno procesiranju slike.



Slika 1.1: Primer strojno izpolnjenega univerzalnega plačilnega naloga, z označenimi območji, ki nas zanimajo za obdelavo.

Za opravljanje storitev z univerzalnim plačilnim nalogom, kot je na primer plačevanje, je seveda iz njega potrebno prebrati podatke. Podobno kot za izpolnjevanje univerzalnega plačilnega naloga, tudi za branje le-tega obstajata ročni in strojni način branja.

Ročni način branja univerzalnega plačilnega naloga je zelo zamudno opravilo, saj mora bralec (zaposleni na banki, pošti, uporabnik spletne banke ...), prebrati vse podatke in jih prepisati v program, ki je temu namenjen. Ponekod, v večjih bančnih poslovalnicah, je število plačevanj in ostalih storitev izvedenih z univerzalnim plačilnim nalogom, zelo veliko in je prepisovanje vseh teh nalogov za zaposlene naporno.

Zato se je tudi pojavila želja po strojnem branju plačilnih nalogov, kjer uporabnik le slika oziroma univerzalni plačilni nalog vstavi v čitalnik, to sliko nato obdela računalnik in uporabniku vrne vse potrebne podatke. Program, ki obdela to sliko, mora torej biti sposoben sprejeti sliko, jo obdelati, prebrati pravilne podatke in jih podati na voljo uporabniku. Posebne naprave (UPN-čitalniki) s programom, ki zna prebrati podatke univerzalnega plačilnega naloga, že obstajajo, vendar kot vhodni podatek dobijo skenirano sliko, ki je vedno isto osvetljena, je lepo poravnana in je na njej le univerzalni plačilni nalog, brez kakršnegakoli ozadja, oziroma drugih stvari. Ti čitalniki delujejo tako, da se omejijo le na spodnji predel (OCR-vrstico) univerzalnega plačilnega naloga in iz njega preberejo podatke. Na podoben način delujejo tudi čitalniki za branje loterijskih listkov [20].

V raziskavi smo zasledili, da je podoben sistem, ki zna prebrati podatke iz univerzalnega plačilnega naloga in jih servirati uporabniku, že na voljo. Nova Kreditna Banka Maribor je v juniju 2012 predstavila svojo mobilno banko, imenovano mBank@Net [4], ki med drugim omogoča tudi skeniranje plačilnih nalogov in s tem enostavno plačevanje položnic.

Uporabniki preprosto skenirajo OCR-vrstico na univerzalnem plačilnem nalogu, aplikacija pa podatke samodejno prenese v obrazec plačilnega naloga. Kako aplikacija zgleda, je prikazano na sliki 1.2.

Slika 1.2: Aplikacija Nove KBM za skeniranje univerzalnega plačilnega naloga.

Bistvena razlika med njihovim sistemom in sistemom, ki ga razvijamo, je v tem, da se oni osredotočijo le na polje znotraj univerzalnega plačilnega naloga, ki vsebuje OCR-zapis, medtem ko bo naš sistem prejel celotno sliko plačilnega naloga. Druga velika razlika je ta, da njihova aplikacija deluje tako, da OCR-polje skenira in ne slika. Mobilno napravo z aplikacijo je potrebno držati nad tem poljem v položaju, določenem z rdečo črto, dokler se ne preberejo vsi znaki, ki so v polju. Naš sistem pa bo deloval tako, da bo uporabnik le slikal plačilni nalog, sliko pa bo nato obdelal program.

Tretja bistvena razlika pa je v namenu sistema. Namen njihovega sistema je omogočiti uporabnikom mobilne banke lažje plačevanje položnic. Poleg mobilne aplikacije, ki zna prebrati univerzalni plačilni nalog, je cilj našega sistema izpolnitev še treh dodatnih nalog.

- Uporaba sistema v aplikaciji spletne banke, v kateri bi s pomočjo spletne kamere slikal plačilni nalog.
- Uporaba kamere na bankomatih, s pomočjo katere bi uporabnik lahko uporabil za slikanje plačilnega naloga in ga tako plačal.
- Zaposlenim na blagajnah v poslovalnicah banke bi omogočili strojno branje univerzalnega plačilnega naloga, tako da bi jim dodali okence s steklom, pod katerim bi bila kamera in na katerega bi nalog le položili in s pritiskom na določen gumb v njihovi aplikaciji sprožili kamero, ki bi nalog slikala, slika pa bi se poslala v obdelavo sistemu.

Glede na določene cilje je naš problem veliko zahtevnejši kot problem razvijalcev Nove KBM pri razvoju njihove aplikacije za branje univerzalnega plačilnega naloga in razvijalcev sistema loterijskega čitalnika. Vhodne slike, ki jih bo dobival naš sistem, so lahko različno zasukane, osvetlitev dveh različnih slik se lahko bistveno razlikuje na slikah, pa so poleg univerzalnega plačilnega naloga, lahko še različna ozadja. Na primer, pri slikanju s spletno kamero in na bankomatu se lahko pojavijo delčki prstov in rok.

Po mojem mnenju so se s podobnimi problemi iskanja določenega polja na sliki in branja njegove vsebine spopadli raziskovalci univerze Calgary, ko so razvijali algoritme za razpoznavanje registrske tablice avtomobilov [5]. Raziskave so se lotili zaradi interesa in povpraševanja po sistemih "slika-radar". Tako kot v našem primeru, je tudi pri njih vhodna slika lahko vedno drugače osvetljena in zasukana, ozadje pa vedno drugo. Razpoznavanja registrske tablice so se lotili v več korakih, predstavljenih v nadaljevanju:

- Predprocesiranje originalne slike: originalno RGB-sliko so najprej pretvorili v sivinsko. V drugem koraku so na sivinski sliki uporabili mediana filter (*angl. median filter*), velikosti 5x5, za odstranitev šuma v sliki ob ohranitvi ostrine slike. Sličica (a), na sliki 1.3, prikazuje primer slike, pretvorjene v sivinsko sliko in z odstranjenim šumom.

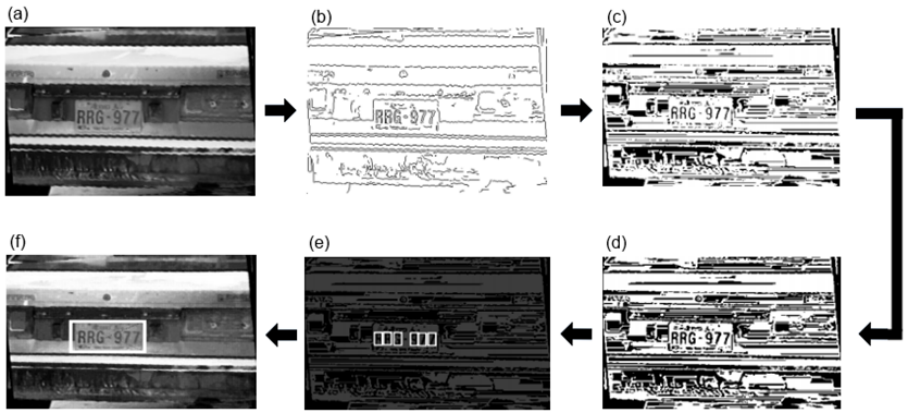
- Detekcija robov: po predprocesiranju slike so poiskali robove s pomočjo Shen-Castanovega detektorja robov. Slika robov (*angl. edge image*) pripomore pri iskanju pravokotnikov v sliki.

- Lociranje znakov: naslednji korak algoritma je iskanje regij slike, na katerih je večja verjetnost, da se nahajajo znaki. Na predprocesirani sliki je najprej izvedeno upragovanje (*angl. threshold*), da sliko spremenijo v binarno obliko. To je najpreprostejša oblika segmentacije oziroma procesa razcepitve digitalne slike v več segmentov, s katero poenostavimo predstavitev slike. Po upragovanju sta na sliki izvedeni še binarna dilatacija (*angl. dilatation*) in erozija (*angl. erosion*), s katerima ločijo med seboj regije v ospredju. Po tem postopku so najdene vse povezane regije v binarni sliki. Iz teh regij so izračunani mejni okvirji (*angl. bounding boxes*). Vsi mejni okvirji z atributi, ki presegajo določene meje, so nato izbrisani. Izbrisani so tudi vsi okvirji, ki ne pripadajo nizom vsaj treh znakov. Če ni najden nobeden okvir, ki ne pripada nizu vsaj treh znakov, je ta meja zmanjšana na dva znaka. Kot rezultat nam ostanejo mejni okvirji, ki so zelo dobri kandidati za znake z registrske tablice.

- Izvedba generičnega algoritma: zadnji korak v njihovi rešitvi je generični algoritem. Ta algoritem, kot vhodni parameter dobi sliko robov in množico mejnih okvirjev kandidatov znakov. Rezultat je osnovan na vsebini slike kotov in lastnostih mejnih okvirjev. V implementaciji vsak kromosom generičnega algoritma predstavlja morebitno pozicijo registrske tablice. Vsak kromosom ima 4 gene, koordinati (x1, y1) za levi zgornji in koordinati (x2, y2) za spodnji kot tablice. Funkcija vrne vrednost med 0 in 1, kjer je 1 odlična registrska tablica, 0 pa slaba tablica. Ta vrednost je izračunana iz ocen registrske tablice štirih kategorij:
  - ocena glede na dimenzijo tablice,
  - ocena glede na to, kako blizu je v robni sliki predstavljen pravokotnik tablice,
  - ocena glede na to, v katerem delu je tablica prekrita z mejnimi okvirji morebitnih črk,
  - ocena glede na to, kako blizu okrog mejnih okvirjev je tablica centrirana; to se pravi, kako dobro so znaki centrirani v tablici.

  Ocene so nato pomnožene še z utežmi pomembnosti, ki jih avtorji določijo sami.

Slika 1.3 prikazuje postopek zgoraj opisane rešitve. Začne se s pretvorbo v sivinsko sliko (a), nato je pognan detektor robov (b), sledi upragovanje (c) in nato še dilatacija in erozija (d). Po predprocesiranju slike je izveden še generični algoritem (e). Na zadnji sličici je prikazan končni rezultat razpoznavanja registrske tablice (f).



Slika 1.3: Postopek algoritma za zaznavanje registrskih tablic avtomobilov.

Naše rešitve optičnega branja univerzalnega plačilnega naloga smo se lotili na zelo podoben način. Podobno kot v opisanem postopku razpoznave registrske tablice, smo program razdelil na štiri korake:

1) predprocesiranje slike (*angl. image preprocessing*),

2) lociranje polj, v katerih je vsebina,

3) priprava najdenih polj na branje,

4) branje vsebine polj.

Poglavje 2

Obdelava slike in branje vsebine

V tem poglavju so opisane uporabljene metode predprocesiranja slike in iskanja polj ter branja vsebine iz njih. Vsaka metoda je najprej predstavljena, nato pa je opisan še način njene uporabe v sistemu.

Za doseganje čim boljših rezultatov branja teksta je potrebno originalno sliko pred samim tekstovnim branjem predhodno obdelati. To dosežemo s postopkom predprocesiranja.

Predprocesiranje [6] je zaporedje oziroma skupina postopkov, s katerimi želimo sliko predelati tako, da s tem povečamo zanesljivost branja in zmanjšamo časovno zahtevnost algoritma za branje teksta. V grobem predprocesiranje delimo na postopke obnove slike (*angl. image restoration*) in postopke izboljšave slike (*angl. image enhancement*).

Obnova slike [7] je operacija, s katero poskušamo iz pokvarjene slike oziroma motne slike z veliko šuma dobiti originalno sliko. Glavni namen obnove slike je, da s čiščenjem slike, s katerim odstranimo šum, pridobimo originalno sliko.

Vizualni šum (*angl. image noise*) [8] je naključna variacija informacij svetlosti in barv v sliki. Povzročen je s strani senzorja in vezja skenerja ali digitalne kamere. Je nezaželen stranski produkt zajema slike, ki sliki doda

lažne in tuje informacije.

Izboljšava slike [7] se od obnavljanja slike razlikuje predvsem v tem, da je njen glavni namen narediti sliko lepšo za opazovalca in poudariti določene lastnosti slike. Pri tem ni nujno, da se pridobi originalno sliko.

Drugi del predprocesiranja [6] pa je izločitev ozadja, črt, okvirjev in podobnih, nepotrebnih elementov v sliki, ki bi lahko povzročili motnje pri razpoznavanju besedila. To lahko storimo s pomočjo detektorjev robov. V primeru, da slika ni lepo poravnana, je del predprocesiranja tudi rotacija slike. Po potrebi pa se sliko v predprocesiranju tudi zgladi, izostri in prilagodi kontrast potrebam detektorja besedila.

Predprocesiranje je v našem algoritmu sestavljeno iz naslednjih korakov:

- pretvorba v sivinsko sliko (*angl. grayscale*),
- obdelava s svetlobnim filtrom,
- detekcija robov (*angl. edge detection*),
- iskanje treh kvadratkov, omenjenih v predstavitvi plačilnega naloga v uvodu in
- rotacija slike.

2.1 Sivinska slika

Prvi korak predprocesiranja slike je pretvorba v sivinsko sliko. Originalno, RGB-sliko najprej pretvorimo v sivinsko sliko.

RGB-slika [9] je slika, predstavljena s tremi kanali, kjer R predstavlja rdečo barvo, G zeleno barvo, B pa modro barvo. Če je slika 24-bitna, ima vsak kanal 8 bitov. Zaradi tega so barvne slike pogosto shranjene kot tri različne slikovne matrike, kjer vsaka matrika predstavlja količino posameznega kanala, oziroma barve v slikovnemu elementu.

Tako kot v barvni RGB-sliki imamo tudi v pretvorjeni, sivinski sliki [10] tri kanale, razlika pa je v tem, da imamo v vseh kanalih enako vrednost.

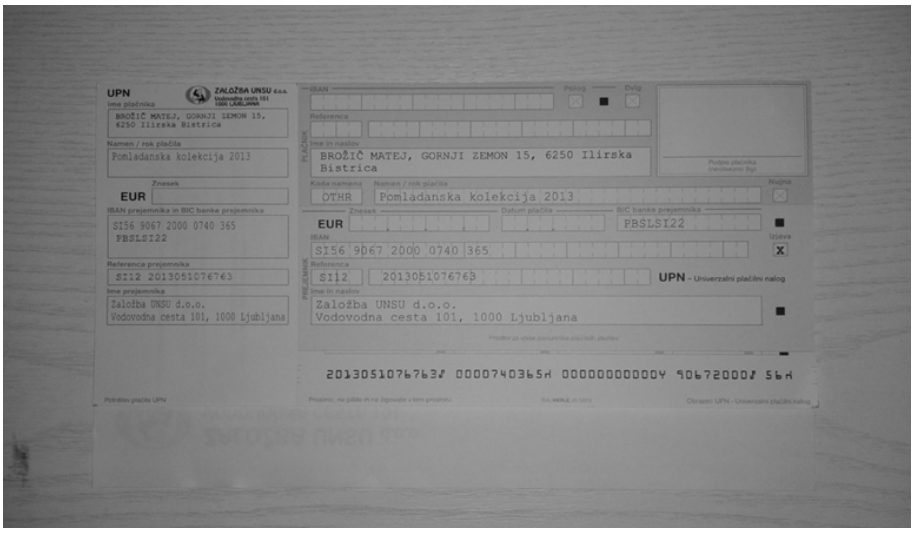
Vrednosti se razlikujejo le v količini svetlobe v posameznem slikovnem elementu. Z več svetlobe dobimo svetlejši slikovni element, z manj svetlobe pa temnejši.

Za pretvorbo barvne slike v sivinsko obstaja več metod, večinoma pa vse delujejo po istem vzorcu. Pred seštevkom posameznih RGB-vrednosti se le te pomnožijo z določenim odstotkom in tako dobimo eno samo vrednost, ki predstavlja svetlost slikovnega elementa. Razlika med metodami je običajno le ta, s kakšno vrednostjo (odstotkom) pomnožimo posamezen kanal in tako dobimo različne svetlosti slikovnega elementa.

V našem sistemu smo za pretvorbo originalne slike v sivinsko uporabili standardno metodo, ki uporablja naslednjo funkcijo za izračun svetlosti slikovnega elementa.

$$S = 0.3R + 0.59G + 0.11B. \tag{2.1}$$

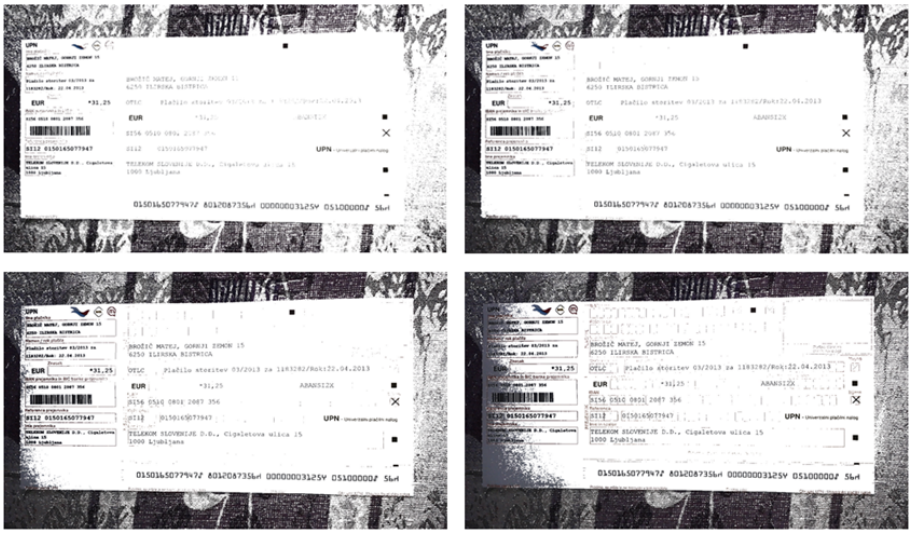
Slika 2.1 prikazuje sivinsko sliko univerzalnega plačilnega naloga, pretvorjeno po omenjeni, standardni metodi.



Slika 2.1: Sivinska slika univerzalnega plačilnega naloga.

2.2 Svetlobni filter

Prvi cilj naše rešitve je seveda zaznava pozicije samega univerzalnega plačilnega naloga na celotni sliki. Pri zaznavi pozicije univerzalnega plačilnega naloga si pomagamo z že omenjenimi tremi črnimi kvadratki, saj z njihovo pozicijo in razdaljo med njimi lahko izračunam širino in višino plačilnega naloga in njegovo dejansko pozicijo, zato jih po pretvorbi originalne RGB-slike v sivinsko sliko poskušamo čim bolj ločiti od ostale slike in ozadja. Pri ločitvi kvadratkov od ozadja je naš prvi korak uporaba svetlobnega filtra nad sliko. Ker so kvadratki med najtemnejšimi elementi na sliki, sliko filtriramo po naslednjem postopku. Iz vseh slikovnih elementih sivinske slike najprej izračunamo povprečno vrednost svetlosti slikovnega elementa. Temu nato prištejemo še štiri različna števila, ki smo jih določili preprosto s poskušanjem in dobimo štiri končne vrednosti. Razlog, da so uporabljene štiri vrednosti, je ta, da kot vhodne parametre v programu dobivamo slike, ki se lahko zelo razlikujejo v svetlosti. S tem se izognemo, da bi filter spregledal kakšno pomembno stvar v sliki. Ko imamo izračunane povprečne vrednosti, se v zanki sprehodimo čez vsak piksel in njegovo vrednost primerjamo z vsako izmed povprečnih vrednosti. Če je vrednost piksla večja od posamezne povprečne vrednosti, ta slikovni element pobarvamo belo. Tako dobimo štiri različne sličice, ki so prikazane na sliki 2.2. Na njih je lepo vidno, kako se slike razlikujejo glede na vrednost, s katero filtriramo.



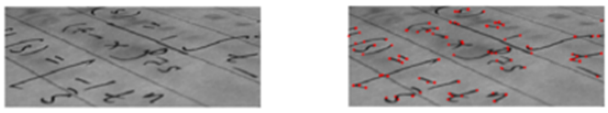
Slika 2.2: Slike, dobljene po uporabi svetlobnega filtra z različnimi povprečnimi vrednostmi.

2.3 Detekcija robov

Ko imamo na voljo slike, obdelane s svetlobnim filtrom, iskanje kvadratov nadaljujemo z metodo detekcije robov. Robovi v sliki nam bodo v veliko pomoč pri iskanju obrisov in s tem tudi iskanih kvadratov.

Detekcija robov [11] je ime za skupek matematičnih metod, katerih cilj je identifikacija točk v digitalni sliki, v katerih se svetlost slike močno spremeni. Točke, kjer je ta cilj dosežen, so tipično združene v skupke točk, ki predstavljajo neke krivulje oziroma črte, imenovane robovi. Detekcija robov je osnovni pojem v procesiranju slik in računalniškem vidu, še posebno, ko govorimo o detekciji in zaznavi elementov v sliki.

Na sliki 2.3 je prikazan rezultat tipičnega detektorja robov.



Slika 2.3: Rezultat detekcije robov.

Obstaja veliko metod za detekcijo robov, ki jih lahko večinoma razdelimo v dve osnovni skupini, detektorji, osnovani na iskanju (*angl. search-based*), in detektorji, osnovani na prehodu ničle (*angl. zero-crossing based*). Detektorji, osnovani na iskanju, najdejo robove tako, da najprej izračunajo mero moči roba in nato iščejo lokalni smerni maksimum magnitude gradienta, z uporabo izračunane ocene lokalne usmerjenosti roba, kar je običajno kar smer gradienta. Gradient je diferencialna operacija, definirana nad skalarnim ali vektorskim poljem, ki pove, v kateri smeri se polje najbolj spreminja.

Detektorji, osnovani na prehodu ničle, temeljijo na iskanju prehoda ničle v izpeljanih izrazih drugega reda, izračunanih iz slike.

Znani detektorji robov se večinoma razlikujejo v tipih glajenja slike, ki jih uporabljajo, in v načinu izračuna mer moči robov. Razlikujejo se tudi v tipih filtrov, uporabljenih za izračun približka gradienta v smeri x-a in y-a. Med najbolj poznanimi detektorji robov so:

- Cannyjev detektor robov (*angl. Canny Edge Detector*),
- Sobelov detektor robov (*angl. Sobel Edge Detector*),
- Prewittov detektor robov (*angl. Prewitt Edge Detector*),
- Robert Crossov detektor robov (*angl. Robert Cross Edge Detector*) in
- Laplacov detektor robov (*angl. Laplace Edge Detector*).

2.3.1 Cannyjev detektor robov

Za detektiranje robov smo se v svojem algoritmu odločili za uporabo Cannyjevega detektorja robov. Razlogi za to so predvsem ti, da je Cannyjev detektor robov v praksi najbolj uporabljan detektor, je zelo neobčutljiv na šum v sliki in v matematičnem smislu velja za optimalni detektor robov.

Cannyjev detektor robov [12, 13] je detekcija robov, ki z uporabo večstopenjskega algoritma detektira široko območje robov v slikah. Razvil ga je John F. Canny. Canny je pri svojem razvoju ciljal na iznajdbo optimalnega

algoritma za detekcijo robov. Optimalni detektor robov bi moral dosegati naslednje:

- dobra detekcija: algoritem mora označiti kar se da veliko pravih robov v originalni sliki,

- dobra lokalizacija: označeni robovi, ki jih najde algoritem, morajo biti kar se da blizu pravih robov v originalni sliki,

- mininalna odzivnost: določen rob v sliki mora biti označen samo enkrat; če je možno, šum v sliki ne sme povzročiti nepravilnih robov.

Za izpolnitev teh pogojev je Canny uporabil tako imenovani izračun variacij (*angl. calculus of variations*). To je metoda, ki najde funkcijo, ki optimizira podano funkcionalnost. Optimalna funkcija v Cannyjevem detektorju je opisana z vsoto štirih eksponentnih pogojev, lahko pa je približana s prvim derivatom Gaussove funkcije [12].

Gaussova funkcija je funkcija, določena z enačbo:

$$f(x) = ae^{-\frac{(a-b)^2}{2c^2}}, \tag{2.2}$$

kjer so a, b in c realna števila in e $\approx$ 2.71828.

Gaussove funkcije so široko uporabljene v statistiki, kjer opisujejo normalne porazdelitve, v procesiranju signalov, kjer so uporabljene za definicijo Gaussovih filtrov, v procesiranju slik, kjer so dvodimenzionalne Gaussove krivulje uporabljene za Gaussovo glajenje, in v matematiki, kjer so uporabljene za reševanje enačb.

Na sliki 2.4 je prikazan primer rezultata Cannyjevega detektorja robov. Leva sličica je originalna slika, na desni pa je prikazana slika robov.

Slika 2.4: Primer Cannyjevega detektorja robov.

2.3.2 Stopnje Cannyjevega detektorja robov

Kot je že omenjeno v prejšnjem poglavju, je Cannyjev detektor robov [12, 13] večstopenjski algoritem. Sestavljen je iz naslednjih stopenj:

- zmanjšanje šuma v sliki (*angl. noise reduction*),
- iskanje gradienta intenzivnosti slike (*angl. finding the intensity gradient of the image*),
- nemaksimalna supresija (odstranjevanje nemaksimalnih vrednosti) (*angl. non maximum suppression*),
- sledenje robovom skozi sliko in histerezno upragovanje (*angl. hysteresis thresholding*).

Za zmanjšanje šuma v sliki Cannyjev detektor robov uporablja filter, osnovan na Gaussovi krivulji, kjer se na originalni sliki izračuna konvolucija z Gaussovim filtrom. Gaussov filter je filter, katerega impulzivni odziv je Gaussova krivulja. Lastnosti Gaussovih filtrov so to, da nimajo nobene prekoračitve vhoda koraka funkcije, medtem ko minimizirajo čas vzpona in padca. Rezultat uporabe Gaussovega filtra na originalni sliki je rahlo zamegljena verzija originala, na katerega nobeden posamezni slikovni element ne vpliva v veliki meri.

Rob v sliki lahko kaže v veliko različnih smereh, zato, za izračun gradienta intenzivnosti, Cannyjev detektor uporablja štiri različne filtre za detekcijo vodoravnih, navpičnih in diagonalnih robov v zamegljeni sliki. Operator detektorja robov vrne vrednost prvega odklona v vodoravni smeri in v navpični

smeri. Iz tega lahko izračunamo gradient roba

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.3}$$

in smer

$$Q = arctan(\frac{G_y}{G_x}) \tag{2.4}$$

Smer roba je zaokrožena na enega izmed štirih kotov, ki predstavljajo vertikalo, horizontalo in dve diagonali.

S podanimi ocenami gradientov slike Cannyjev detektor nato nadaljuje s preverjanjem, ali magnituda gradienta zavzame lokalen maksimum v smeri gradienta. Ta postopek se imenuje nemaksimalna surpresija. Z njim dobimo skupek točk, prikazan kot binarna slika.

Zadnji korak Cannyjevega detektorja je sledenje robov. Gradienti z visoko intenzivnostjo bolj verjetno ustrezajo robovom kot gradienti z nizko intenzivnostjo. V večini primerov je nemogoče določiti prag, v katerem gradient s podano intenzivnostjo ustreza robovu in v katerem ne. Zato Canny v svojem algoritmu uporablja upragovanje s histerezo. Upragovanje s histerezo zahteva dva praga – visokega in nizkega. Začnemo z uporabo visokega praga, ki nam označi robove za katere je velika verjetnost, da so pravi. Tem robovom nato sledimo skozi sliko. Ko jim sledimo, uporabimo še nizek prag, ki nam omogoča, da sledimo robovom tudi skozi svetle in zbledele dele slike, dokler ne najdemo začetne točke. Po koncu tega procesa imamo binarno sliko, na kateri je vsak slikovni element označen kot rob ali nerob. Dobljeno binarno sliko lahko obravnavamo tudi kot skupek robnih krivulj.

2.3.3 Pomembni algoritmi Cannyja

Cannyjev algoritem [12, 13] vsebuje številne različne, nastavljive parametre, ki imajo vpliv na čas, ki je potreben za izračun robov, in na uspešnost detektorja. Najbolj pomembna sta dva.

- Velikost Gaussovega filtra - filter za glajenje slike, uporabljen v prvem koraku, ima zelo velik vpliv na rezultat Cannyjevega algoritma. Manjši filtri povzročijo manjšo zameglitev in omogočajo detekcijo manjših ostrih črt. Večji filtri povzročijo večjo zameglitev slike in so boljši za detekcijo večjih, bolj gladkih robov.

- Pragovi - uporaba dveh pragov s histerezo dovoljuje večjo fleksibilnost kot pristop z enim samim pragom, še vedno pa veljajo standardni problemi upragovanja. Previsok prag lahko spusti pomembne informacije, po drugi strani pa lahko prenizek prag nepomembne informacije napačno obravnava kot pomembne. Zelo težko je podati neki standardni prag, ki bi deloval za vse slike.

Povzamemo lahko, da je Cannyjev detektor robov zelo prilagodljiv različnim okoljim, saj njegovi parametri dovoljujejo, da se ga prilagodi zaznavi robov različnih karakteristik in slik.

2.3.4 Uporaba Cannyjevega detektorja

Po pretvorbi slike v sivinsko sliko in uporabi svetlobnega filtra, s katerim dobimo štiri slike, v našem algoritmu sledi detekcija robov z uporabo Cannyjevega detektorja robov. Cannyjev detektor uporabimo nad vsemi štirimi slikami, ki jih dobimo po uporabi svetlobnega filtra. Kot rezultat dobimo štiri binarne slike, ki jih lahko vidimo na sliki 2.5.

Slika 2.5: Slike, dobljene z uporabo Cannyjevega detektorja robov v našem sistemu.

2.4 Dilatacija in erozija

V prvotno načrtovani rešitvi našega algoritma smo po uporabi Cannyjevega detektorja robov nameravali uporabiti iskalnik obrisov in z njim najti obrise v sliki, ampak se to ni izkazalo kot dovolj dobra rešitev. Kot rezultate smo namreč tako dobili ponekod preveč kvadratov, ponekod pa premalo. Da bi prišli do boljših rezultatov pri iskanju kvadratov, smo morali slike, ki jih dobimo s Cannyjevim detektorjem, nekako izboljšati. To smo dosegli z morfološkim procesiranjem slike.

Morfološko procesiranje [14] je procesiranje binarnih slik, sestavljeno iz operacij, s katerimi je objekt X spremenjen s strani strukturiranega objekta B, pri čemer dobimo obliko, ki je bolj ustrezna za obdelavo prepoznavanja vzorcev. Elementa v medsebojni interakciji (X in B) sta predstavljena kot skupka v evklidskem dvodimenzionalnem prostoru (*angl. euclidean dimensional space*).

Evklidski prostor je realni topološki vektorski prostor, v katerem je definiran skalarni produkt. S pomočjo skalarnega produkta lahko potem v evklidskem prostoru merimo razdalje in kote.

Večina morfoloških operacij je izpeljana iz dveh osnovnih morfoloških operacij, dilatacije (*angl. dilation*) in erozije (*angl. erosion*).

Dilatacija, z drugo besedo tudi razširjenje, poskuša zapolniti manjše vdore in s tem razširiti regijo na sliki. Definicija: dilatacija X z B, označena z X$\ominus$B, je množica vseh točk x, kjer imata Bx in X neprazna presečišča:

$$X \ominus B = X|B_x \subset X \tag{2.5}$$

Če poenostavimo, je dilatacija operacija, ki vsak slikovni element ozadja, ki se dotika objekta, spremeni v slikovni element objekta.

Erozija, z drugo besedo tudi slabitev, poskuša odstraniti manjše vdore in s tem oslabiti oziroma stanjšati regijo na sliki. Definicija: erozija X z B, označena z X$\oplus$B, je množica vseh točk x, kjer je Bx vključen v X:

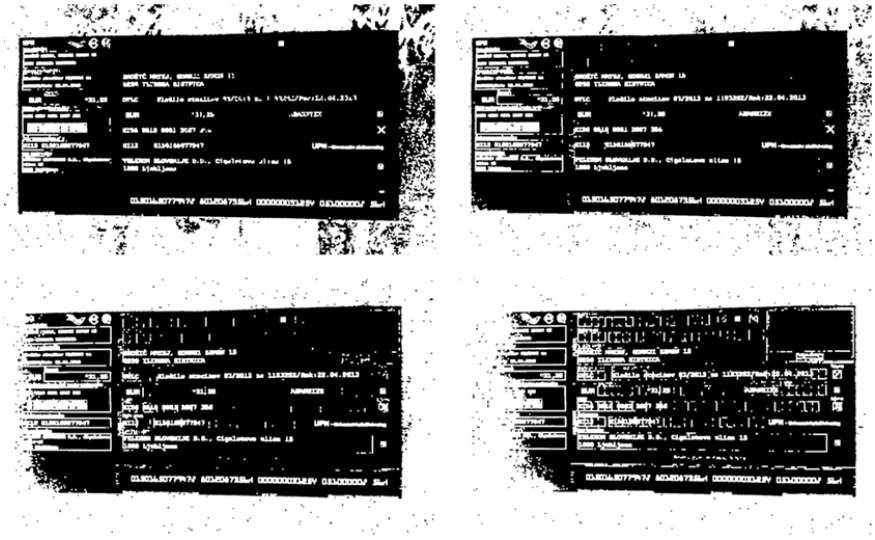
$$X \oplus B = X|B_x \cap X \neq 0 \tag{2.6}$$

Če poenostavimo, je erozija operacija, ki vsak slikovni element objekta, ki se dotika ozadja, spremeni v slikovni element ozadja.

Na sliki 2.6, je prikazan dober primer erozije in dilatacije. Leva sličica je originalna slika, na sredini je prikazana erozija te slike, na desni sličici pa vidimo dilatacijo prve, originalne slike.

Slika 2.6: Dilatacija in erozija.

V našem algoritmu smo s pomočjo erozije in dilatacije zmanjšali število zaznanih obrisov in s tem izboljšali zaznavo kvadratov. Na binarnih slikah, ki jih dobimo s Cannyjevim detektorjem robov, je najprej uporabljena dilatacija, s katero zapolnemo morebitne luknje v zaznanih robovih, nato pa uporabimo še erozijo, da z dilatacijo odebeljene robove nazaj stanjšamo. Slike, dobljene s tema dvema operacijama, so prikazane na sliki 2.7.



Slika 2.7: Slike robov po dilataciji in eroziji.

2.5 Iskanje in detekcija kvadratkov

Po korakih pretvorbe v sivinsko sliko, svetlobnem filtru, detekciji robov in dilataciji ter eroziji, imamo štiri binarne slike robov, na katerih lahko brez večjih problemov razpoznamo kvadratke. Detekcijo in razpoznavo kvadratkov implementiramo tako, da najprej uporabimo iskanje obrisov, ki najde vse obrise v sliki, ki pa jih nato filtriramo tako, da ostanejo samo želeni kvadratki.

Naslednji korak je torej detekcija okvirjev v sliki. Programsko rešitev smo razvili v programskem jeziku C++ s pomočjo knjižnice OpenCV [15].

OpenCV je odprtokodna knjižnica za delo z računalniškim vidom. Napisana je v jeziku C in C++, poganja pa se lahko na operacijskih sistemih Linux, Windows in Mac OS X. Mogoče jo je vključiti tudi v druge programske jezike.

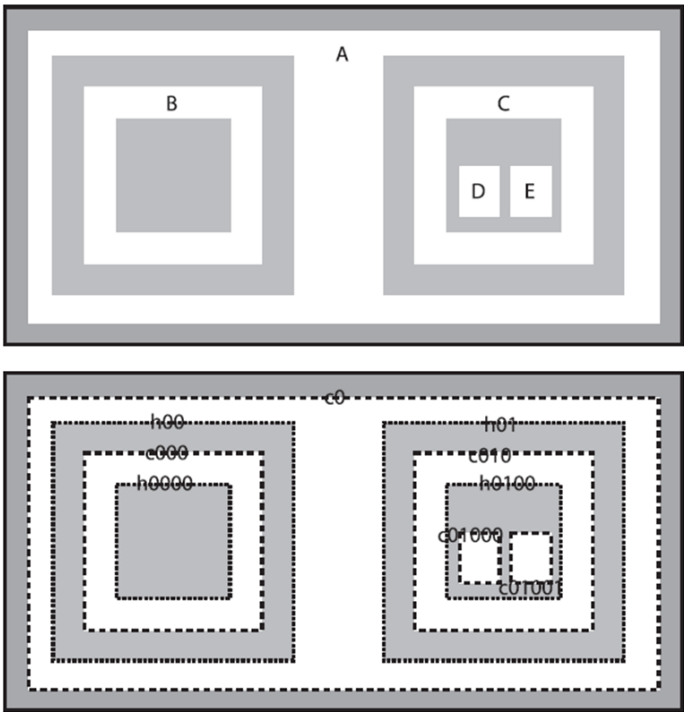
Razvita je bila za računalniško učinkovitost in z velikim poudarkom na aplikacije v realnem času. Napisana je v optimiziranem jeziku C in zna uporabljati večjedrne procesorje. Glavni cilj knjižnice OpenCV je zagotoviti enostavno infrastrukturo računalniškega vida (*angl. computer vision*), ki uporabnikom pomaga hitro zgraditi zapletene aplikacije, ki uporabljajo računalniški vid. Knjižnica vsebuje več kot 500 metod za delo z računalniškim vidom. Ker je računalniški vid velikokrat povezan s strojnim učenjem (*angl. machine learning*), vsebuje OpenCV tudi podporo za strojno učenje.

Za iskanje obrisov je v našem sistemu uporabljena metoda iz knjižnice OpenCV, imenovana `cvFindContours()`. Preden gremo v podrobnosti metode, si poglejmo, kaj sploh je obris. Obris je seznam točk, ki predstavlja neko povezano krivuljo v sliki. Ta predstavitev se lahko razlikuje glede na trenutne okoliščine v sliki. V knjižnici OpenCv so obrisi predstavljeni z zaporedji, v katerih vsak vnos v zaporedje kodira informacijo o lokaciji naslednje točke v krivulji. Ta zaporedja so predstavljena z objektom `cvSeq` (povezan seznam struktur).

2.5.1 Iskanje obrisov

Metoda `cvFindContours()` [15] poišče oziroma izračuna obrise iz binarnih slik, zato je bilo potrebno slike najprej obdelati s Cannyjevim detektorjem.

Funkcionalnost metode `cvFindContours()` prikazuje slika 2.8. Na zgornjem delu slike je prikazana testna slika, ki vsebuje več belih območij (označenih od A do E) na temnem ozadju. Na spodnjem delu pa je upodobljena ista slika, na kateri so označeni obrisi, ki jih najde metoda `cvFindContours()`. Ti obrisi so označeni s cX ali hX, kjer c pomeni obris (*angl. contour*), h pomeni luknja (*angl. hole*), X pa je neko število. Nekateri obrisi so prikazani s prekinjenimi črtami. To so zunanje meje belih območij. Drugi pa so prikazani s pikastimi črtami in so notranje meje oziroma zunanje meje lukenj (temnih območij).



Slika 2.8: Testna slika podana metodi `cvFindContours()`.

OpenCV omogoča tudi razvrščanje najdenih obrisov v drevo obrisov, ki

zakodira relacije vsebovanosti v svojo strukturo. Drevo obrisov, ki bi ustrezalo zgornji testni sliki, bi imelo obris c0, kot svoje korensko vozlišče. Luknji h00 in h01 bi bili njegova otroka, ta pa bi naprej imela obrise, ki jih direktno vsebujeta kot svoje otroke, in tako naprej.

Metodi `cvFindContours()` je potrebno podati šest parametrov in kot rezultat vrne število vseh najdenih obrisov (int), predstavljena pa je na naslednji način:

```
int cvFindContours(
 IplImage*               img,
 CvMemStorage            storage,
 CvSeq*                  firstContour,
 int                     headerSize,
 CvContourRetrevialMode  mode,
 CvChainApproxMethod     method
);
```

Prvi parameter je vhodna slika, ki naj bi bila 8-bitna slika z enim kanalom in je interpretirana kot binarna (vsi neničelni slikovni elementi so enaki med seboj). Med izvajanjem metoda `cvFindContours()` uporabi to sliko kot prostor za računanje, zato je potrebno narediti kopijo slike in podati to kopijo metodi, če želimo sliko uporabiti pozneje.

Naslednji parameter, `storage`, označuje prostor, v katerem lahko metoda najde prostor v spominu, v katerega shrani obrise. Ta prostor je potrebno pred uporabo rezervirati. V knjižnici OpenCV se to stori z metodo `cvCreateMemStorage()`, ki ustvari entiteto, imenovano `memory storage`, ki je namenjena ravnanju s spominskim prostorom za dinamične objekte.

Naslednji je `firstContour`, ki je kazalec na `CvSeq*`. Metoda `cvFindContours()` ga rezervira sama, zato da ga uporabniku ni potrebno. Namesto tega pošljemo v metodo le kazalec na `firstContour`, da ga lahko metoda nastavi. V tem objektu (`firstContour`) lahko najdemo kazalec na krovni element izdelanega drevesa obrisov.

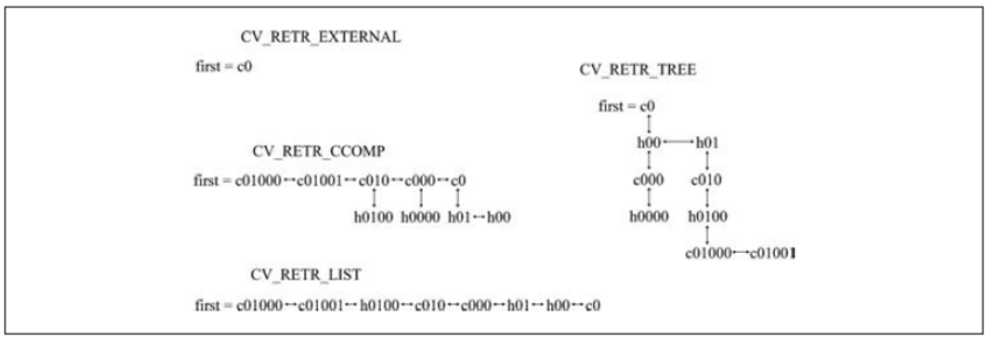
Parameter `headerSize` metodi samo pove več o objektih, ki bodo rezervirani. Lahko je nastavljeno na `sizeof(CvContour)` ali na `sizeof(CvChain)`.

Zadnje je uporabljeno, ko je metoda približevanja nastavljena na `CV_CHAIN_CODE`.

Naslednja sta na vrsti parametra `mode` in `method`, ki nam povesta več o tem, kaj natančno naj metoda računa in kako naj to računa.

Parameter »mode« je lahko nastavljen na eno izmed štirih možnosti: `CV_RETR_EXTERNAL`, `CV_RETR_LIST`, `CV_RETR_CCOMP`, ali `CV_RETR_TREE`. Vrednost parametra `mode` pove metodi, točno katere obrise želimo najti in kako bi radi, da nam je rezultat predstavljen.

Na sliki 2.9 so prikazane topologije (*angl. topologies*), ki jih dobimo kot rezultat vsake izmed štirih možnih vrednosti za način, ki ga določa paramater `mode`. V vsakem izmed primerov si strukture lahko predstavljamo kot ravni, ki so povezane z vodoravnimi povezavami (`h_next` in `h_prev`) in ki so ločene med seboj z navpičnimi povezavami (`v_next` in `v_prev`).



Slika 2.9: Prikaz različnih vrednosti parametra `mode` v metodi `cvFindContours()`, glede na vhodno sliko, prikazano na sliki 2.8.

Način CV_RETR_EXTERNAL

Vrne samo najbolj zunanje obrise. V sliki 2.8 imamo samo en zunanji obris, zato je v sliki 2.9 za to vrednost prikazan samo prvi obris (c0), ki kaže na to zaporedje točk, drugih povezav pa ni prikazanih.

Način CV_RETR_LIST

Vrne vse obstoječe obrise in jih doda v seznam. Slika 2.9 prikazuje seznam obrisov, za to vrednost, ki jih dobimo kot rezultat za vhodno sliko na sliki 2.8. V tem primeru je najdenih osem obrisov, ki so vsi med seboj povezani

s `h_prev` in `h_next`.

Način CV_RETR_CCOMP

Vrne vse obstoječe obrise in jih organizira v dvonivojsko hierarhijo, kjer so meje na prvem nivoju vse zunanje meje komponent, meje na drugem nivoju pa so meje vseh lukenj. Glede na sliko 2.9 lahko vidimo, da imamo za vhodno sliko s slike 2.8 pet zunanjih mej, od katerih tri meje vsebujejo luknje. Luknje so povezane s pripadajočimi zunanjimi mejami z `v_next` in `v_prev`. Najbolj zunanja meja c0 vsebuje dve luknji. Ker ima lahko `v_next` samo eno vrednost, ima lahko vozlišče samo enega otroka. Vse luknje, znotraj c0, so zato povezane ena z drugo s `h_prev` in `h_next` kazalci.

Način CV_RETR_TREE

Vrne vse obstoječe obrise in rekonstruira celotno hierarhijo ugnezdenih obrisov. V našem primeru (sliki 2.8 in 2.9) to pomeni, da je najbolj zunanji obris c0 korensko vozlišče. Pod njim je luknja h00, ki je povezana s še eno luknjo h01 na istem nivoju. Vsaka od teh lukenj ima otroke (obrisa c000 in c010), ki so povezani s svojimi starši z vertikalnimi povezavami. Tako se povezovanje nadaljuje do najbolj notranjih obrisov v sliki, ki predstavljajo zadnja vozlišča (liste) v drevesu.

Naslednji parameter v metodi je `method`, ki pove, kako so obrisi predstavljeni. Lahko ima pet vrednosti, opisanih spodaj.

Najprej si poglejmo, kaj je verižna koda (*angl. chain code*) [16]. Verižna koda je brezizgubni stiskalni algoritem (*angl. lossless compression algorithm*) za enobarvne slike (slike, ki so v celoti naslikane z eno barvo oziroma več odtenki ene barve). Osnovni princip verižne kode je, da posamezno zakodira vsako povezano komponento v sliki. Za vsako takšno regijo je nato izbrana točka na meji in njene koordinate so posredovane. Kodirnik se nato pomika po meji regije in v vsakem koraku posreduje simbol, ki predstavlja smer njegovega gibanja. Tako se nadaljuje, dokler se kodirnik ne vrne nazaj na svoj začetni položaj. Na tej točki je bila trenutna povezana komponenta v celoti opisana in kodirnik nadaljuje z naslednjo. Ena najbolj znanih verižnih kod je Freemanova.

Metoda CV_CHAIN_CODE

Obrisi, ki jih vrne `cvFindContours()`, so shranjeni kot Freemanove verižne kode (*angl. Freeman chain code*). Vse ostale vrednosti vrnejo poligone (zaporedja točk).

Metoda CV_CHAIN_APPROX_NONE

Spremeni oziroma pretvori vse točke iz verižne kode v točke.

Metoda CV_CHAIN_APPROX_SIMPLE

Stisne (*angl. compresses*) vodoravne, navpične in diagonalne segmente. Izpusti le njihove končne točke.

Metoda CV_CHAIN_APPROX_TC89_L1

Uporabi Teh-Chinov verižni algoritem za približke.

Metoda CV_LINK_RUNS

Uporablja se popolnoma drugačen algoritem od prej naštetih, ki poveže vodoravne segmente. To metodo lahko uporabimo samo, če je parameter `mode` nastavljen na `CV_RETR_LIST`.

2.5.2 Uporaba metode in filtriranje obrisov

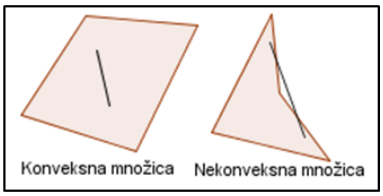
Metoda `cvFindContours()` je v našem sistemu uporabljena na vsaki izmed štirih binarnih slik, ki jih dobimo v predhodnem koraku (detekcija robov). Parametra metoda (*angl. method*) in način (*angl. mode*), ki sta podana metodi, sta `CV_RETR_LIST`, ki vrne seznam vseh obstoječih obrisov in `CV_CHAIN_APPROX_SIMPLE`, ki iz rezultata izloči vse vmesne točke in pusti samo točke na robu. Tako dobimo, na primer, za štirikotnik samo štiri robne točke, ki predstavljajo njegov obris.

Obrise, ki jih dobimo za posamezno sliko, nato dodamo v seznam (`CvSeq*`) in tako dobimo v enem seznamu obrise vseh štirih binarnih slik. Poleg obrisov želenih treh kvadratkov pa dobimo v seznam seveda tudi vse ostale obrise, nekateri obrisi pa so tudi podvojeni, saj združimo skupaj vse obrise štirih slik. Naslednji korak sistema je zato filtriranje dobljenih obrisov.

Prvi filter je implementiran že pri dodajanju obrisov posamezne slike v omenjeni skupni seznam. Preden je obris dodan v skupni seznam, mora

izpolnjevati določene pogoje. Dodan je le, če:

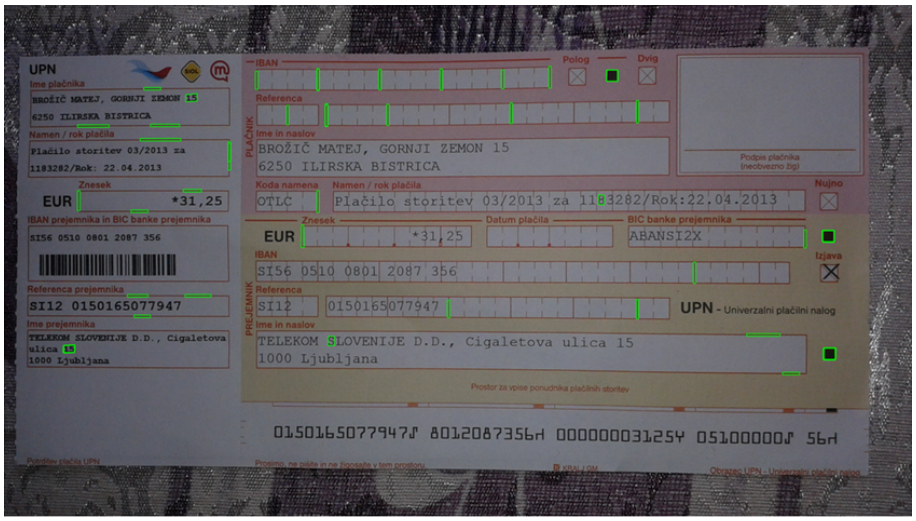
- ima natančno štiri točke, ker nas zanimajo samo štirikotniki,
- je konveksen (vsi robovi so izbočeni); primer lahko vidimo na sliki 2.10,



Slika 2.10: Primer konveksnosti in nekonveksnosti.

- njegova velikost ustreza določeni meji in
- vsi njegovi koti so večji od 90 stopinj; to velja takrat, ko za vsak kot obrisa velja, da je njegov sinus zelo majhen.

Obrisi, ki jih dobimo z uporabo prvega filtra nad vsemi dobljenimi obrisi, so prikazani na sliki 2.11.



Slika 2.11: Obrisi, ki jih dobimo po prvem filtriranju.

Kot je razvidno že iz zgornje slike, rezultati dobljeni s prvim filtriranjem niso dovolj dobri, da bi si pomagali z njimi, saj je število obrisov še vedno preveliko, nekateri enaki oziroma zelo podobni obrisi pa so v seznamu zapisani tudi dvakrat.

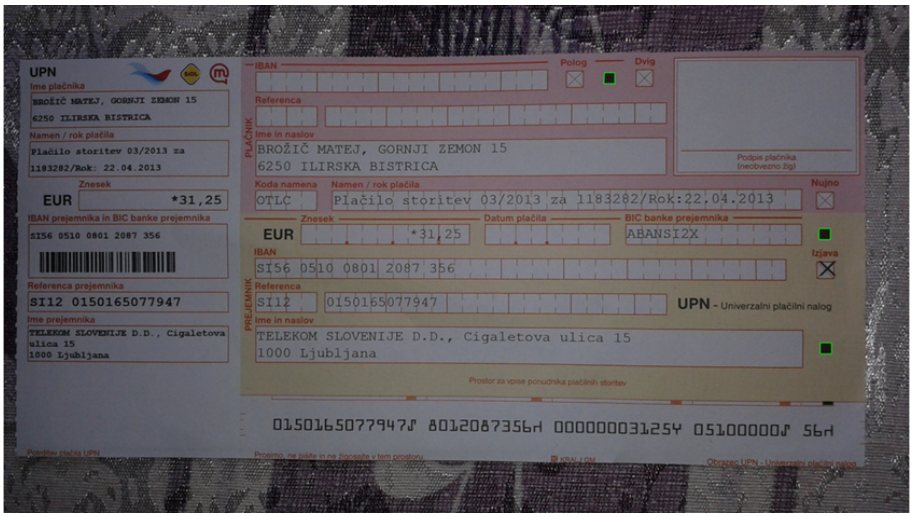
Zato je potrebno implementirati še dodatno filtriranje. Drugo filtriranje je izvedeno nad seznamom obrisov, dobljenim s prvim filtriranjem. Za vsak obris iz tega seznama se izvede naslednje preverjanje:

- lokacija obrisa - obris se mora nahajati na desni polovici celotne slike,
- ozadje obrisa - ozadje obrisa mora biti precej temnejše od ozadja v okolici obrisa,
- razmerje stranic obrisa - stranici obrisa a in b morata biti v razmerju 1 : 1; ta pogoj je izpolnjen takrat, ko za stranici a in b velja:
  abs(a - b) <= 2 pikslov.
- unikatnost obrisa - v seznamu ne sme biti dveh enakih, oziroma zelo podobnih obrisov; o preverimo tako, da vsak obris iz seznama primerjamo z vsemi ostalimi obrisi; če v seznamu ne obstaja nobeden drugi

obris, ki ima približno enak obseg in se hkrati nahaja na približno enaki lokaciji, potem je obris unikaten.

Obrise, ki izpolnjujejo vse zgornje pogoje, obdržimo, v nasprotnem primeru pa so izbrisani iz seznama.

Po drugem filtriranju dobimo želene tri obrise kvadratkov, s katerimi lahko določimo pozicije polj, ki jih iščemo. Prikazani so na sliki 2.12.



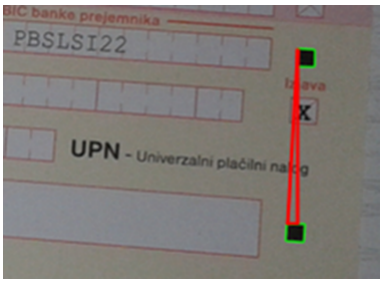
Slika 2.12: Obrisi, ki nam ostanejo po uporabi drugega filtra.

2.6 Rotacija slike

V aplikacijah, s katerimi bodo uporabniki zajemali slike univerzalnega plačilnega naloga, bo sicer v naprej določena pozicija, kako naj nalog stoji, vseeno pa bo dovoljeno manjše odstopanje pri slikanju. Zato lahko na vhodu pričakujemo tudi nekoliko zasukane slike. Za poravnanje takšne slike je potrebno implementirati rotacijo.

Ali je podana slika kandidat za rotacijo, preverimo s pomočjo treh najdenih kvadratkov v predhodnem koraku. Najprej pregledamo pozicije vseh treh kvadratkov na sliki in se osredotočimo na dva kvadratka, ki sta najbolj desno. Ta dva kvadratka morata biti točno eden nad drugim, zato lahko z njuno pomočjo izračunamo kot zasuka slike.

Najprej odštejemo y-koordinato leve zgornje točke zgornjega kvadrata od leve zgornje točke spodnjega kvadrata in tako dobimo razdaljo navpične črte med njima. Nato isto ponovimo še za x-koordinato in dobimo razdaljo vodoravne črte med točkama. S tema dvema razdaljama lahko sedaj z uporabo Pitagorovega izreka izračunamo še dejansko razdaljo med točkama in tako dobimo pravokotni trikotnik, prikazan na sliki 2.13.



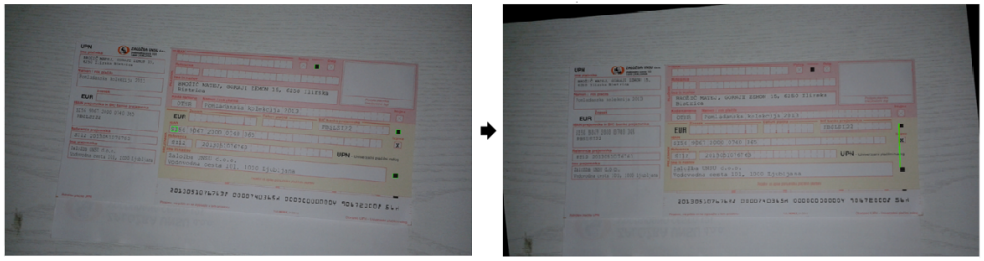
Slika 2.13: Pravokotni trikotnik, dobljen s pomočjo razdalj med točkama.

Kot zasuka slike je enak zgornjemu kotu izračunanega pravokotnega trikotnika. Izračunamo ga s formulo za izračun kota v pravokotnih trikotnikih:

$$\cos \alpha = \frac{prileznaKateta}{hipotenuza} \tag{2.7}$$

Sliko rotiramo samo v primeru, če je ta kot večji od dveh stopinj, saj je v nasprotnem primeru slika dovolj dobra in je ni potrebno rotirati.

Na sliki 2.14 je prikazan primer rotacije slike, ki je izvedena v sistemu, po zgoraj opisanem postopku.



Slika 2.14: Rezultat rotacije slike.

2.6.1 Iskanje polj UPN-ja

Po detekciji kvadratkov in rotaciji imamo sedaj na voljo pravilno rotirano sliko, z označenimi tremi kvadratki. Imamo torej dovolj podatkov, da določimo, kje v slikanem univerzalnem plačilnem nalogu se nahajajo polja z vsebino, ki nas zanimajo.

Kot je omenjeno že v uvodu, iščemo polji Koda namena in Namen/Rok plačila v zgornjem delu naloga, polja Znesek, IBAN in Referenca v sredinskem delu in vrstico OCR, ki je čisto na dnu naloga (vsa ta polja so označena na sliki 1.1).

Preden se lotimo iskanja polj, je potrebno izračunati višino in širino plačilnega naloga (samo dela ki nas zanima, brez talona za stranko). Širino izračunamo tako, da razdaljo osi $\boldsymbol{x}$ (vodoravne osi), med zgornjim in sredinskim kvadratkom, pomnožimo s 3.05, višino pa tako, da razdaljo osi $\boldsymbol{y}$ (navpične osi), med sredinskim in spodnjim kvadratkom, pomnožimo z 2.7.

$$sirina = (sredinskiKvadrat \rightarrow x - zgornjiKvadrat \rightarrow x) * 3.05 \quad (2.8)$$

$$visina = (spodnjiKvadrat \rightarrow y - sredinskiKvadrat \rightarrow y) * 2.7 \quad (2.9)$$

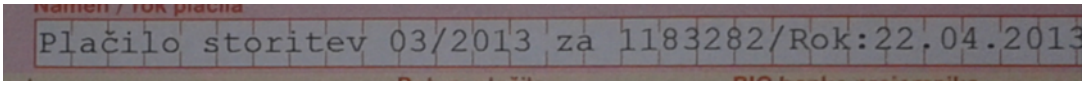
Polje: Koda namena

Prvo polje, ki ga program poišče, je Koda namena. Njegovo pozicijo dobimo tako, da vzamemo sredinski kvadratek, višino in širino, in s pomočjo teh vrednosti izračunamo oddaljenost polja od tega kvadratka. Rezultat iskanja je prikazan na spodnji sliki 2.15.

Slika 2.15: Zaznano polje, ki vsebuje kodo namena.

Polje: Namen/Rok plačila

Naslednje polje je na vrsti Namen/rok plačila. Njegovo pozicijo dobimo tako, da prvemu polju, ki ga že imamo zaznanega, prištejemo dolžino stranice enega od kvadratkov. Dobljeni rezultat prikazuje slika 2.16.



Slika 2.16: Zaznano polje, ki vsebuje Namen/rok plačila.

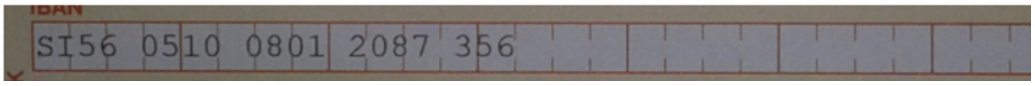
Polje: Znesek

Pozicijo polja zneska, podobno kot pri Kodi namena, dobimo tako, da vzamemo sredinski kvadratek, višino ter širino, in izračunamo, kje se nahaja. Na sliki 2.17 je prikazano polje, ki ga dobimo.

Slika 2.17: Zaznano polje, ki vsebuje Znesek.

Polje: IBAN

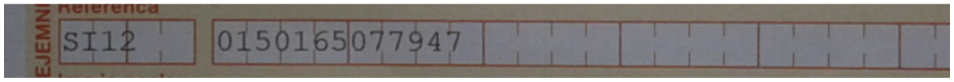
Polje z IBAN-om se nahaja bliže spodnjemu kvadratku, zato tokrat namesto srednjega kvadratka vzamemo spodnjega. Tako kot pri Kodi namena in Znesku tudi tukaj pozicijo izračunamo s širino in višino naloga. Polje, ki ga s tem dobimo, je prikazano na sliki 2.18.



Slika 2.18: Zaznano polje, ki vsebuje IBAN.

Polje: Referenca

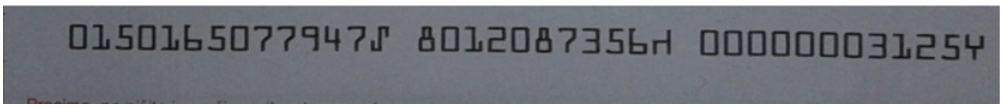
Predzadnje polje, ki ga poiščemo, je polje Referenca. Vodoravno se začne v isti točki kot polje IBAN, zato vzamemo za pozicijo osi x isto točko kot za IBAN in s pomočjo spodnjega kvadratka izračunamo le, kje na osi y se nahaja. Polje je prikazano na sliki 2.19.



Slika 2.19: Zaznano polje, ki vsebuje Referenco.

Polje: OCR-vrstica

Ostane nam še zadnje iskanje polja OCR-vrstice. Poiščemo jo tako, da enako kot pri referenci za pozicijo osi x vzamemo isto točko kot za IBAN in poiščemo še pozicijo na osi y. Na sliki 2.20 je prikazana OCR-vrstica.



Slika 2.20: Zaznana OCR-vrstica.

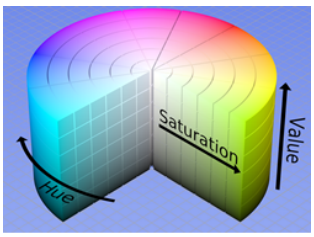
2.7 Razpoznava besedila (OCR)

Za postopkom predprocesiranja, s katerim smo dobili polja, ki jih potrebujemo za branje besedila znotraj njih, je seveda na vrsti branje tega besedila. Branje besedila je sestavljeno iz dveh delov. Drugi del branja je optično razpoznavanje znakov (*angl. optical character recognition - OCR*), s katerim so prebrani posamezni znaki. Ker pa optično razpoznavanje znakov daje mnogo boljše rezultate, če je vhodna slika črno bela in lepo obdelana (odstranjeni robovi, ozadje, črte ...), je prvi korak predpriprava zaznanih polj.

2.7.1 Priprava zaznanih polj na branje

Cilj priprave na branje je torej odstranitev čim več motečih objektov v ozadju, kot so robovi, črte in ostale stvari, ki bi lahko zmotile OCR pri branju. Če si podrobno ogledamo zaznana polja, lahko opazimo, da je ozadje precej svetlejše od pisave, robovi in črte, ki označujejo polja, pa so precej rdeče barve.

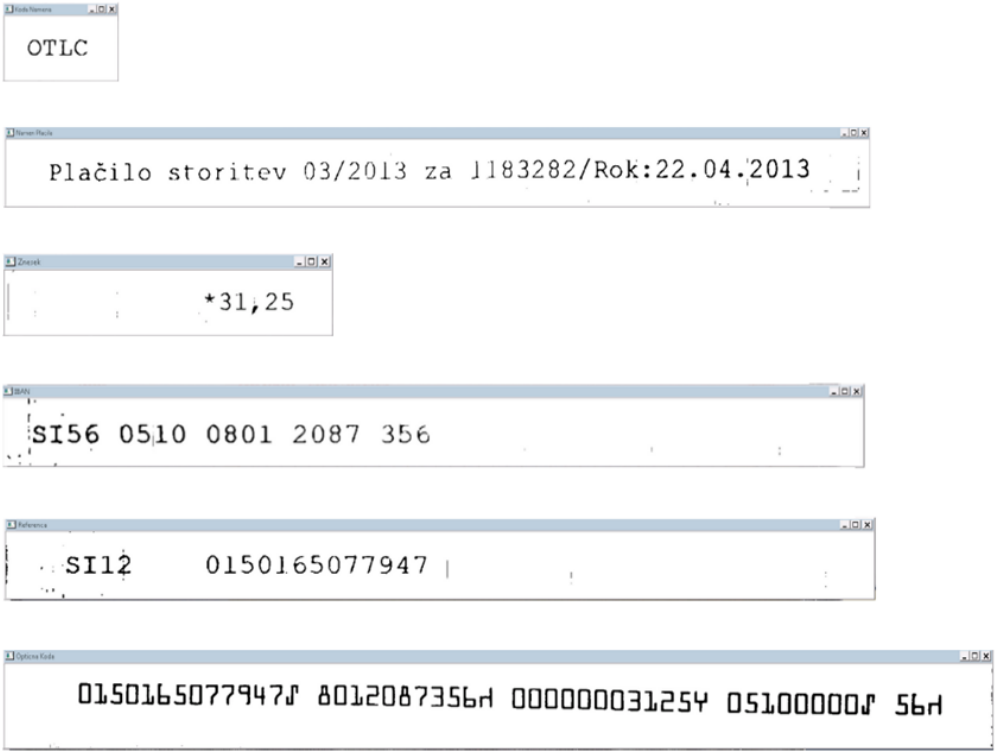
Glavni del priprave polj je zato uporaba rdečega filtra, ki slikovne elemente, ki presegajo določen prag rdeče barve, spremeni v slikovne elemente bele barve. Slika je v barvnem prostoru RGB, zato jo najprej pretvorimo v barvni prostor HSV (Slika 2.21), v katerem je lažje izolirati določeno barvo, ki jo želimo. Tako kot pri RGB-slikah, tudi HSV hrani barvne informacije v treh kanalih, le da tukaj kanal V (Value) drži informacijo o svetlosti, ostala dva kanala pa vsebujeta informaciji o barvi.



Slika 2.21: Model, ki predstavlja HSV barvni prostor.

Filter gre čez vse slikovne elemente v sliki in primerja vrednosti H, S in V z določenimi pragi. Če katera od vrednosti presega prag, se ta slikovni element pobarva v belo, v nasprotnem primeru pa se pobarva v črno. Tako se znebimo vseh rdečih črt in robov polj, ki bi lahko motili OCR. Hkrati se v belo prebarva tudi ozadje, saj se preverja tudi vrednost V, ki predstavlja svetlost. Tako dobimo slike, ki imajo le črne in bele slikovne elemente in jih OCR dosti lažje prebere.

Ponekod v slikah lahko ostane še kakšen šum, zato na sliki uporabimo še glajenje, s katerim odstranimo morebitni šum. Tako dobimo iz polj na slikah, prikazanih na prejšnjih straneh, pripravljena polja na branje vsebine, ki so prikazana na sliki 2.22.



Slika 2.22: Zaznana polja, pripravljena na OCR-branje.

2.7.2 Optično branje besedila v poljih

Z vhodne slike, ki jo imamo na začetku, smo s pomočjo predprocesiranja in zaznave polj dobili dele slike, na katerih so polja, ki nas zanimajo. Ta polja smo nato še obdelali s svetlobnim filtrom ter jih zgladili. Sedaj je na vrsti še samo optično razpoznavanje znakov v zaznanih poljih.

Optično razpoznavanje znakov [17, 18, 19] je mehanična ali elektronska pretvorba skeniranih slik ročno napisanega ali natipkanega besedila oziroma znakov v strojno kodiran tekst. Pogosto se uporablja kot način vnosa podatkov iz neke vrste originalnih podatkov papirnatega izvora (dokumenti, pošta, poljubni tiskani zapisi). Je pogosta metoda digitalizacije tiskanih besedil, tako da se lahko elektronsko išče po njih, so boljše shranjena, prikazana na spletu in uporabljena v strojnih procesih, kot na primer strojno prevajanje besedila in rudarjenje besedila (postopek pridobivanja kakovostnih informacij iz besedila).

Zgodnje verzije je bilo potrebno programirati s slikami vsakega znaka in so delovale za posamezno pisavo naenkrat. Sodobnejši sistemi, ki so na voljo sedaj, so inteligentni in imajo visoko stopnjo natančnosti prepoznave večine pisav. Nekateri sistemi so sposobni reproducirati formatirane izpise, ki so skoraj enaki kot originalna skenirana stran [16].

Obstaja veliko plačljivih in odprtokodnih sistemov za optično razpoznavanje znakov. Med najbolj znanimi so:

- Tesseract (odptrokodni),
- ABBY (plačljiv),
- IRIS ReadIRIS (plačljiv),
- Nuance (plačljiv)

Pri odločanju, kateri sistem OCR uporabiti za branje, so najpomembnejše lastnosti, na katere je treba biti pozoren, naslednje:

- natančnost prepoznavanja znakov,

- natančnost pri rekonstrukciji izpisa strani,
- podpora za razne jezike in pisave,
- hitrost.

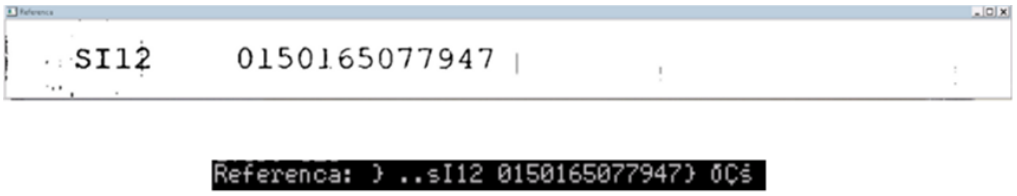
Po raziskavi raznih sistemov OCR in pregledovanju njihovih lastnosti in zmogljivosti smo se odločili za uporabo Tesseract-a. Temeljni razlog je ta, da je odprtokoden in hkrati zelo zmogljiv.

Tesseract

Tesseract [20] je najverjetneje nabolj natančen odprtokodni sistem. Zmožen je prebrati veliko različnih formatov slik in jih pretvoriti v besedilo v več kot 60 različnih jezikih, možno pa ga je naučiti tudi nov jezik oziroma pisavo. V testu natančnosti UNLV, leta 1995, je bil med prvimi tremi sistemi. Sistemu je bil med leti 1995 in 2006 zelo malo izboljšan, po tem pa je njegov razvoj prevzel Google, ki je do danes uvedel veliko optimizacij.

Tesseract deluje na operacijskih sistemih Linux, Windows in Mac OSX, lahko pa je uporabljen tudi na ostalih sistemih, kot na primer Android in iPhone, vendar le-ti niso dobro testirani.

Sistemu Tesseract smo sprva za test kar podali svoja predobdelana polja z besedilom s testne slike, brez kakršnekoli predpriprave. Rezultati, ki smo jih dobili, so bili presenetljivo dobri, razen vrstice OCR, kjer je posebna pisava EUROBANKING, ki je Tesseract ni znal prebrati. Tesseract je v približno 80 % znake prebral pravilno, kar je zelo dobro. Izjeme so bile predvsem znaki »a«, ki jih je sistem zamenjal z »@«; znaki »S«, ki jih je sistem zamenjal s »5« in podobno. Druge izjeme so bile, da je sistem Tesseract šum slike prepoznal kot znake. Primer branja polja Referenca s sistemom Tesseract je prikazan na sliki 2.23. Tesseract na vhod dobi polje Referenca, kot rezultat pa nam vrne nabor znakov, ki jih prepozna.



Slika 2.23: Branje polja Referenca s sistemom Tesseract.

Vidimo lahko, da so vsi znaki na sliki prepoznani pravilno. Problem je v tem, da so poleg njih prepoznani še nekateri znaki, ki jih na sliki ni. Zato smo se odločili, da sistem Tesseract še izboljšamo in ga naučimo nov jezik oziroma pisavo.

Učenje Tesseracta

Zadnja verzija Tesseracta (verzija 3.2) je naučena prebrati več kot 60 različnih jezikov. Če pa uporabniku to ni dovolj, lahko sistem nauči nov jezik. V verziji 2 je bilo učenje Tesseracta kar zapleten proces, ki je vzel veliko časa. Z verzijo 3 in naprej pa so razvijalci razvili Tesseract Trainer, ki učenje zelo poenostavi.

Za potrebe sistema optičnega branja univerzalnega plačilnega naloga smo Tesseract naučili dve novi pisavi. Najprej smo ga naučil pisave Courier New, z omejenim naborom znakov. Dovoljeni znaki so le vse slovenske črke, številke od 0 do 9, vejica (,), pika (.), poševnica (/), dvopičje (:), podpičje (;), minus (-) in zvezdica (*). To so vsi znaki, ki jih lahko pričakujemo v poljih univerzalnega plačilnega naloga, zato drugih ne potrebujemo in bi nas samo motili.

Kot drugo smo ga naučil pisavo, s katero je napisana vsebina v vrstici OCR. To je pisava OCR-A1 EUROBANKING. To je posebna pisava, ki je uporabljena samo besedilo v OCR-vrstici in je namenjena prav boljši prepoznavi z optičnim branjem znakov.

Postopek učenja sistema Tesseract, pisave OCR-A1 EUROBANKING smo izvedli v več korakih.

1. korak: Namestitev potrebnih programov

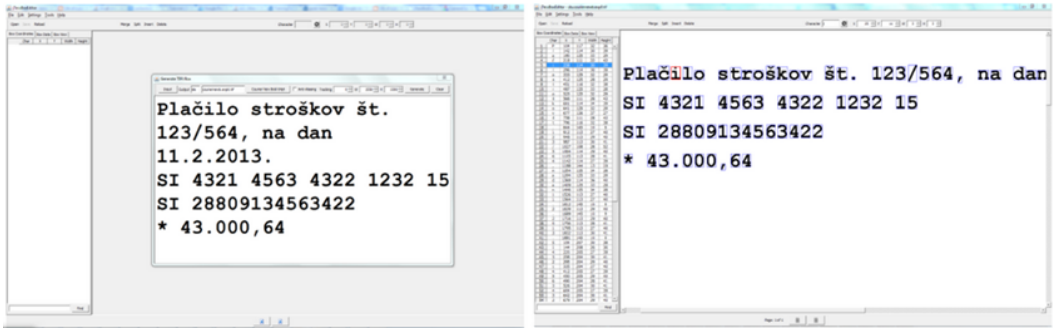
Preden se sploh lotimo učenja nove pisave, je seveda potrebno namestiti programe, ki jih za to potrebujemo. Namestili smo programa

- Tesseract 3.2 Trainer (program za učenje Tesseracta) in
- JTessBox Editor (program za izdelavo slik in datoteko okvirjev (*angl. box file*)).

2. korak: Izdelava slike in datoteke z okvirji

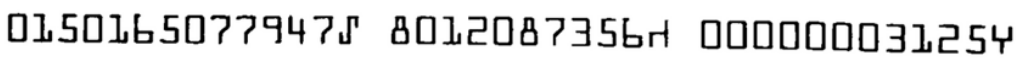
Program JTessBoxEditor omogoča zelo enostavno izdelavo slik in datotek z okvirji, saj ima urejevalnik, kjer se opravi vse, kar je potrebno za izdelavo le-teh.

Za izdelavo slike in datoteke z okvirji za učenje pisave Courier New je v urejevalniku le potrebno nastaviti ustrezno pisavo, velikost pisave in razmak med znaki ter izbrati opcijo "generiraj". S tem izdelamo sliko in okvirje za prvo pisavo. Program JTessBoxEditor, njegov urejevalnik s primerom vsebine in rezultat izdelave slike pisave so prikazani na sliki 2.24. Na levi strani slike je prikazan urejevalnik programa JTessBoxEditor z besedilom, na desni strani pa vidimo sliko in ustrezne okvirje, ki jih program izdela iz besedila na levi strani.



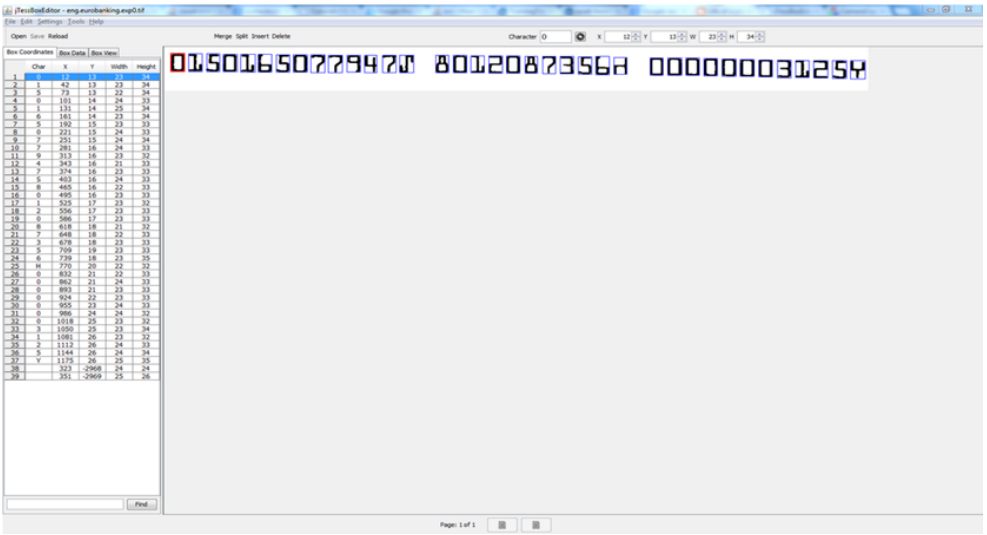
Slika 2.24: Urejevalnik jBoxText Editor z vnesenim besedilom in rezultat generiranja slike in okvirjev.

Pisave OCR-A1 EUROBANKING pa v urejevalniku programa ni, zato smo sliko in datoteko z okvirji izdelali ročno. Sliko smo izdelali tako, da smo iz testne slike univerzalnega plačilnega naloga izrezali vrstico OCR in jo spremenili v črno-belo sliko (slika 2.25). Pomembno je, da shranimo sliko kot datoteko TIFF, saj je Tesseract možno učiti le s tem formatom slik. Slika vsebuje nabor vseh možnih znakov v OCR-vrstici, zato je dovolj dobra za učenje Tesseracta.



Slika 2.25: Slika formata TIFF, ki jo uporabimo za učenje Tesseracta pisave OCR-A1 EUROBANKING.

Sedaj je sliki potrebno dodati še ustrezne okvirje. To smo storili tako, da smo v datoteki okvirjev za sliko pisave Courier New spremenili njene vrednosti, da ustrezajo sliki EUROBANKING pisave. Rezultat (slike in okvirji), ki ga tako dobimo, je prikazan na sliki 2.26.



Slika 2.26: Slika pisave OCR-A1 EUROBANKING z označenimi okvirji.

Poglavje 3

Testiranje in rezultati

V tem poglavju je predstavljeno testiranje in rezultati sistema, opisanega v prejšnjem poglavju. Testiranja smo se lotili v štirih delih in z dvema načinoma zajema slik. Najprej nas je zanimala pravilnost zaznave treh kvadratkov in s tem pozicije univerzalnega plačilnega naloga na sliki. Nato smo preverili pravilnost rotacije, po tem še pravilnost računanja pozicij polj in pravilnost branja vsebine v poljih.

Prvi način zajema slik, ki je bil uporabljen, je zajem z mobilnimi napravami. Uporabljene mobilne naprave so bile Samsung Galaxy S3, Samsung Galaxy S2 (sistem Android), Samsung Galaxy S2(sistem Windows Mobile), Samsung Galaxy ACE2, Nokia Xperia, Nokia XpressMusic 5800, Nexus 3 in Iphone 4s.

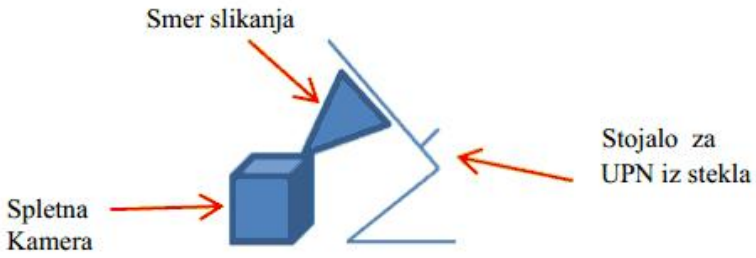
Z vsako napravo je bilo zajetih 20 slik različnih ločljivosti, razen s Samsung Galaxy Ace2, s katero je bilo zajetih 40 slik. Pri zajemanju slik so veljala naslednja navodila:

- dovoljen je le manjši naklon (do 10 stopinj),
- slikamo v več različnih prostorih in na več različnih mestih znotraj enega prostora (pri oknu, stran od okna),
- slikamo v različnih obdobjih dneva pri različnih osvetlitvah (jutro, večer, popoldan),

- slikamo nekaj slik z bliskavico, nekaj brez,
- slikamo pri različnih virih osvetlitve (rumena, bela),
- slikamo na podlagah različnih barv in tekstur.

Tako smo dobili 180 različnih slik z različnimi osvetlitvami, ozadji, nakloni in ločljivostmi.

Drugi način zajemanja slik pa je potekal z uporabo spletne kamere. V primeru, če bi se odločili sistem implementirati na blagajne v poslovalnicah Banke Koper, bi to storili s pomočjo spletne kamere, na način, prikazan na sliki 3.1.



Slika 3.1: Spletna kamera in stojalo za zajem slike univerzalnega plačilnega naloga.

Za testiranje smo stojalo sestavili iz dveh škatel (ena večja, ena manjša) in steklenega okenca, ki smo ga postavili na škatli. Vmes smo postavili spletno kamero "Logitech Quickcam Pro 9000 Wired USB PC Webcam", ki je bila priklopljena na testni računalnik.

Za razliko zajemanja slik z mobilnimi napravami, kjer smo slike zajeli pred testiranjem, so bile s spletno kamero slike zajete ob samem testiranju, tako da se je po zajetju slike takoj sprožil naš sistem. Na ta način smo zajeli in testirali 100 različnih univerzalnih plačilnih nalogov. Za razliko od slik, zajetih z mobilnimi napravami, so bile te slike vse pravilno poravnane, slikane v istem prostoru in na isti svetlobi.

Skupaj je bilo torej z našim sistemom testiranih 280 različnih slik univerzalnih plačilnih nalogov. Testiranje je potekalo v štirih korakih.

3.1 Testiranje zaznave položaja univerzalnega plačilnega naloga

Prvi test je bil test prve naloge našega sistema - zaznave treh črnih kvadratkov. Testirala se je pravilnost iskanja in filtracije obrisov. Pozicije polj z vsebino, ki jih iščemo, so najboljše izračunane, če imamo vse tri kvadratke, možno pa jih je izračunati že z lokacijo dveh kvadratkov. Glede na to smo določili, da je naš prvi test uspešen, če sta zaznana najmanj dva kvadratka in največ tri kvadratki.

Test zaznave položaja je potekal delno avtomatsko. V primeru zajema slik z mobilnimi aparati smo vse slike shranili znotraj ene mape. Te slike je sistem rekurzivno bral in v primeru, da je našel dva ali tri kvadratke, je povečal določen števec. V primeru, da je sistem našel dva ali tri kvadratke, smo ročno pregledali, če so bili najdeni kvadratki pravilni.

V primeru zajema slik s spletno kamero, kjer je testiranje potekalo takoj za zajemom slike, pa smo sistemu dodatno določili, da slike, kjer je uspešno zaznal dva ali tri kvadratke, shrani znotraj dveh map, ki smo ju ustvarili za uspešen test, v nasprotnem primeru pa znotraj mape za neuspešen test. Tudi tukaj smo pravilnost najdenih kvadratkov preverili ročno.

3.2 Testiranje rotacije slik

Drugi del testiranja je bil test pravilnosti rotacije slik, ki niso pravilno poravnane. Izmed vseh 180 slik je bilo za rotacijo primernih 20 slik z različnimi nakloni. Ostale slike so bile že na vhodu pravilno poravnane. Kriterij, ki smo ga določili za uspešnost testa, je velikost kota v pravokotnem trikotniku, dobljenim z desnima kvadratkoma (opisano v poglavju 2.6) po rotaciji. Test je bil uspešen v primeru, ko je bil ta kot manjši od dveh stopinj. Preverjala

se je tudi pravilnost izračuna prvotnega naklona (če je naklon manjši od dveh stopinj, se rotacija ne sme izvesti).

Ta del testiranja, je edini potekal v celoti avtomatsko. V primeru zajema slik z mobilnimi aparati je sistem rekurzivno bral slike in je, če je bila rotacija uspešna, povečal določen števec.

Testiranje pravilnosti rotacije slik zajetih s spletno kamero, je potekalo na enak način kot pri prvem testiranju, le da se je upošteval drugačen kriterij in nam slike ni bilo treba preverjati še ročno.

3.3 Testiranje računanja pozicije polj

Tretji test je bil test računanja pozicije šestih polj z vsebino. Za razliko od prvih dveh testiranj ta del testiranja ni potekal avtomatsko, ampak je bil v celoti izveden ročno. Uspešnost testa smo ocenili tako, da smo ročno pregledali sličice polj, ki jih je naš sistem izračunal za posamezno sliko.

Za oba načina zajemanja slik je testiranje potekalo na enak način. Za vsako sliko smo preverili rezultate računanja pozicij (dobljene sličice), in če so bila na sličicah lepo razvidna vsa polja in vsebina znotraj njih, je bil test uspešen.

3.4 Testiranje branja vsebine

Četrti in zadnji del testiranja je bil test pravilnosti branja vsebine v izračunanih poljih. Tako kot tretji del testiranja je tudi ta test v celoti potekal ročno in ga nismo avtomatizirali. Kriterij uspešnosti tega testa je bila pravilnost prebrane vsebine, glede na izvorno vsebino v poljih.

Testiranje je potekalo tako, da smo za vsako vhodno sliko, ne glede na njen način zajema slike, primerjali rezultat branja vsakega izmed izračunanih polj z originalno vsebino. Primerjali smo ročno, tako da smo prebrali originalno vsebino s slike in naše rezultate ter primerjali, če se ujemata. Če se vsebini ujemata, je ta test uspešen.

3.5 Optimizacija testiranja

Zaradi različnih izboljšav sistema, ki zahtevajo veliko ponovnih testiranj, je testiranje na zgoraj opisani način, zelo časovno potratno in naporno. V izvajanju so že optimizacije, s katerimi nameravamo testiranje našega sistema popolnoma avtomatizirati, kar bo ponovne teste občutno skrajšalo, saj ne bo potrebnega ročnega pregledovanja slik.

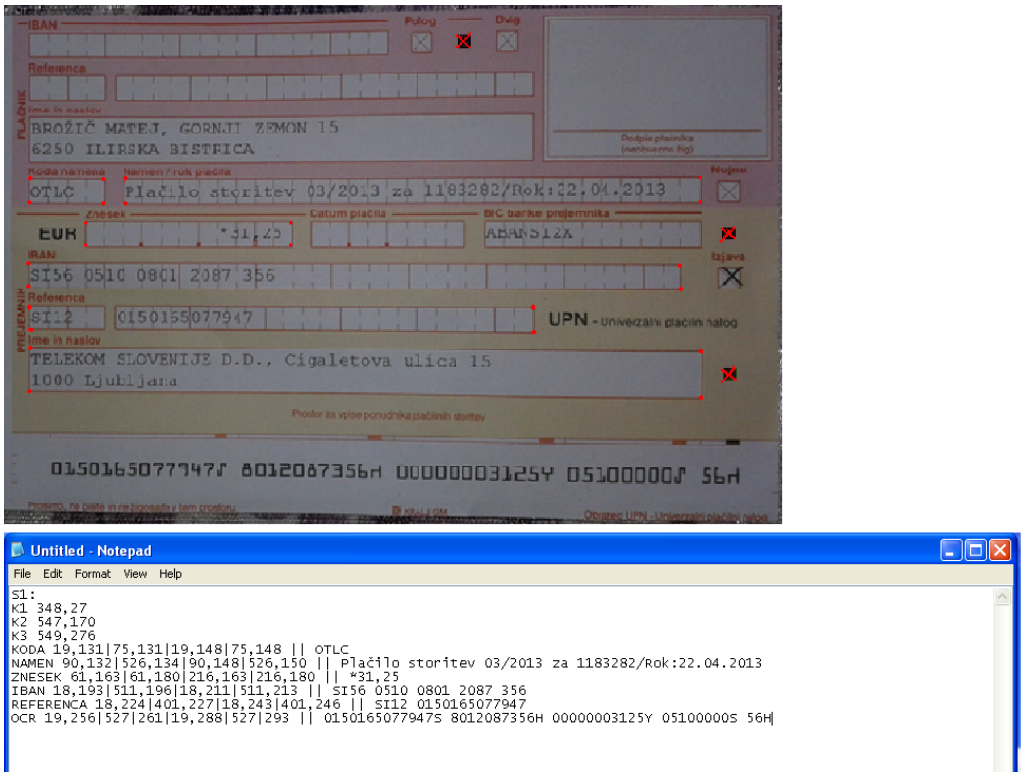
Za avtomatizacijo testiranja je za prvi test potrebno na vsaki izmed slik določiti center vsakega izmed treh kvadratkov, s katerim bomo lahko nato preverjali rezultate, ki jih vrne naš sistem. Testiranje pravilnosti rotacije je že v celoti avtomatizirano, zato ni potrebnih novih izboljšav.

Za potrebe našega tretjega testiranja (test izračuna pozicije polj), je na vsaki izmed slik potrebno določiti točke štirih oglišč vsakega polja. Te točke bomo nato v testih primerjali s pozicijami, ki nam jih vrne naš sistem.

Zadnje testiranje bo avtomatizirano tako, da bo iz vsake izmed slik prebrana vsebina polj, ki bo v testu primerjana z vsebino, ki jo vrne naš sistem.

Vse opisane vrednosti bodo zapisane v tekstovni datoteki, ki bo vključena k slikam in jo bo pri testiranju sistem prebral skupaj s sliko. Pomembno je, da pri primerjanju rezultatov našega sistema z vrednostmi v datoteki upoštevamo določeno napako, ki bo določala, kdaj je test uspešen in kdaj ne.

Primer takšne datoteke je prikazan na sliki 3.2. Zgoraj je slika univerzalnega plačilnega naloga, na kateri so označene točke centrov kvadratkov in oglišča polj, spodaj pa je tekstovna datoteka s koordinatami teh točk in vsebino v poljih.



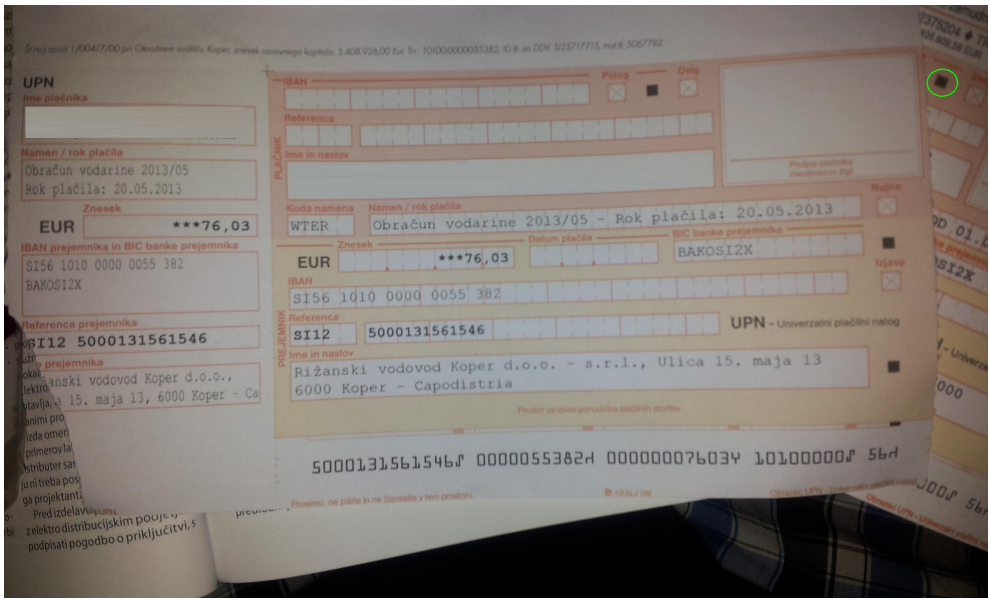
Slika 3.2: Nalog z označenimi centri kvadratkov in oglišči polj ter tekstovna datoteka z vrednostmi točk in vsebino v poljih.

3.6 Rezultati testiranja

Tako kot testiranje samo so tudi rezultati le-tega predstavljeni v štirih delih.

3.6.1 Pravilna zaznava položaja

Prvi testi je bil torej test pravilnosti zaznave treh kvadratkov. Izmed vseh 180 slik, zajetih z mobilnimi napravami, je naš sistem pravilno zaznal tri kvadratke v 166 in dva kvadratka v 7 primerih. Pri sedmih slikah torej ta test ni bil uspešen. Pri iskanju vzroka napačne zaznave smo ugotovili, da je problem pri svetlobnem filtru, opisanem v poglavju 2.2. Vse slike, kjer je bil test neuspešen, so imele črno ozadje. Povprečna vrednost in s tem tudi filtracija v svetlobnem filtru je bila zato napačna. Svetlobni filter smo zato prilagodili tako, da je namesto v celotni sliki povprečje po novem računal le na sredinskem delu slike. Tako se ozadje ni upoštevalo. Po naši prilagoditvi je sistem za 179 z mobilnimi napravami zajetih slik pravilno zaznal položaj univerzalnega plačilnega naloga na sliki (3 kvadratke na 173 slikah in 2 kvadratka na 6 slikah). Edina slika, na kateri je sistem zaznal preveč kvadratkov, je prikazana na sliki 3.3.



Slika 3.3: Slika z zaznanimi štirimi kvadratki.

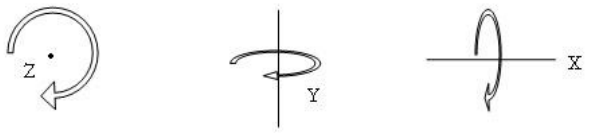
Razlog za napačno zaznavo je dodaten kvadratek na sliki (obkrožen z zeleno barvo), zaradi katerega sistem namesto treh kvadratkov zazna štiri, saj dodaten kvadratek izpolnjuje vse pogoje, ki jih preverjamo z našima filtroma. Rešitev tega problema, ki jo nameravamo vključiti v sistem, je dodatno preverjanje pozicije posameznega kvadratka glede na ostale kvadratke. Dva izmed kvadratkov se nahajata eden nad drugim (s tem, da je potrebno upoštevati možnost nepravilno poravnane slike), tretji kvadratek pa mora biti nad obema in levo od njiju. Druga dodatna kontrola pa je razdalja med kvadratki.

V primeru zajemanja slik s spletno kamero je sistem pred našo prilagoditvijo in po njej pravilno zaznal vse tri kvadratke pri vseh 100 slikah.

3.6.2 Pravilnost rotacije

Pri testiranju pravilnosti rotacije slike so bili rezultati sledeči. Izmed 20 nepravilno poravnanih slik je bilo 18 pravilno rotiranih in dve nepravilno. Izmed ostalih 160 testnih slik, ki so bile pravilno poravnane, je sistem pravilno

izpustil in ni rotiral 152 slik, pri 8 slikah pa je narobe izračunal velikost kota naklona slike. Razlog za nepravilno rotacijo in nepravilen izračun velikosti kota naklona je, da pri izračunu upoštevamo le naklon glede na os z oziroma sredinsko točko slike (levo na sliki 3.4), ne upoštevamo pa naklon glede na os x (desno na sliki 3.4) in naklon glede na os y (na sredini slike 3.4).



Slika 3.4: Različni načini rotacije slike.

Glede na to, da nameravamo že z aplikacijami, v katerih bo vključen naš sistem, uporabnika usmeriti k pravilni poravnavi slike, smo z rezultati tega testiranja zadovoljni, zato zaenkrat ni bilo uvedenih nobenih izboljšav rotacije. Dopolnitev, ki jo nameravamo dodati v prihodnosti je zaznava zunanjih robov plačilnega naloga in z njimi izračunati perspektivo ter s pomočjo le te nato pravilno poravnati sliko.

3.6.3 Računanje položaja polj

Pri tretjem testu, testu pravilnosti računanja položajev polj, je naš sistem za slike, zajete s spletno kamero, pravilno izračunal položaje polj pri vseh 100 slikah, za slike, zajete z mobilnimi napravami, pa je napačno izračunal položaje le za eno sliko (če odštejemo sliko, na kateri je sistem napačno zaznal kvadratke), pri vseh ostalih pa so bili položaji pravilno izračunani. Slika, na kateri je sistem napačno izračunal položaje, je bila ena izmed šestih slik, na kateri sta bila najdena le dva kvadratka, in edina slika, kjer sta to bila levi kvadratek zgoraj in desni kvadratek spodaj. Razlog za napačne pozicije polj je po vsej verjetnosti napačen izračun le teh s tema dvema kvadratkoma, vendar pa napake zaenkrat še nismo odkrili.

Rezultati tega testa so bili za nas sprejemnljivi in nismo uvedli nobenih izboljšav na tem področju.

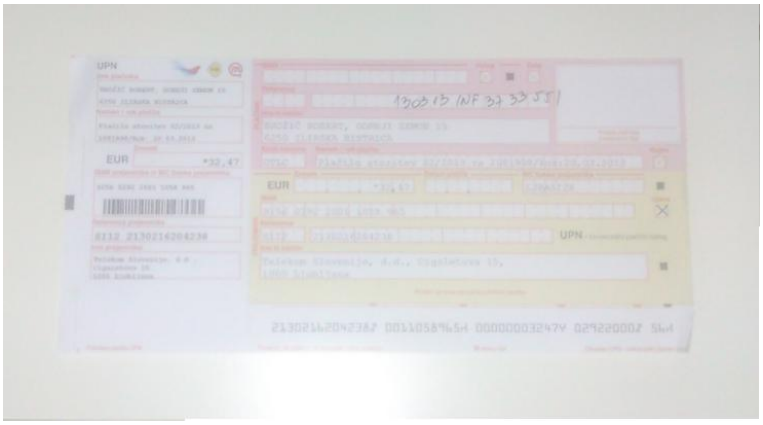
3.6.4 Branje vsebine

Rezultate branja vsebine smo preverjali v dveh delih.

OCR-vrstica

Najprej smo se lotili testiranja branja najbolj spodnjega polja, OCR-vrstice, ki je po našem mnenju najpomembnejše polje. Izmed 100 slik, zajetih s spletno kamero, je sistem pravilno pripravil na branje in prebral 92 slik. Pri slikah, zajetih z mobilnimi napravami, pa je bilo pravilno prebranih 159 slik. Rezultati nas niso prepričali, zato smo naš sistem poskusili izboljšati.

Glede na to, da je polje OCR-vrstica bolj ali manj le črna pisava na beli podlagi, smo se odločili preizkusiti branje vsebine na polju brez predobdelave. Rezultati so bili precej boljši. Po našem preizkusu je sistem pravilno prebral vse s spletno kamero zajete slike, od 180 slik, zajetih z mobilnimi napravami, pa je napačno prebral le 6 slik, ki so vsebovale kar nekaj šuma in so bile že na pogled težko berljive. Primer takšne slike je prikazan na sliki 3.5.



Slika 3.5: Primer slike, z veliko šuma, kjer sistem ne zna prebrati vsebine OCR vrstice.

Ostala polja

Naslednje je bilo na vrsti preverjanje ostalih polj. Rezultati tega testiranja so predstavljeni v spodnji tabeli.

Ime Polja	Način zajema	Vseh slik	Pravilnih	%
Koda namena	Mobilna naprava	180	161	89,44
	Spletna kamera	100	94	94,00
Namen plačila	Mobilna naprava	180	139	77,22
	Spletna kamera	100	84	84,00
Znesek	Mobilna naprava	180	152	84,44
	Spletna kamera	100	93	93,00
IBAN	Mobilna naprava	180	133	73,89
	Spletna kamera	100	90	90,00
Referenca	Mobilna naprava	180	130	72,22
	Spletna kamera	100	88	88,00

Tabela 3.1: Rezultati branja vsebine iz polj univerzalnega plačilnega naloga.

Da bi izboljšali rezultate, smo predobdelavo pred samim branjem razdelili na dva dela, predobdelavo za slike, zajete s spletno kamero, in predobdelavo za slike, zajete z mobilnimi napravami, ter vsako predobdelavo prilagodili po svoje. Poleg tega smo sistem Tesseract OCR naučili 5 novih jezikov ter vsako polje prebrali s svojim jezikom.

Za polje koda namena obstaja 125 različnih vrednosti, zato smo Tesseract naučili nov jezik s sliko, ki je vsebovala vseh 125 vrednosti.

Slike za učenje Tesseracta novih jezikov za boljše branje ostalih štirih polj - namen plačila, znesek, IBAN in referenca, smo izdelali tako, da smo vzeli vsebine teh polj iz 50 naključnih univerzalnih plačilnih nalogov ter iz njih izdelali slike za učenje Teseracta.

S tema optimizacijama smo uspeli branje slik, zajetih s spletno kamero, precej izboljšati. Malenkost nam je uspelo izboljšati tudi slike, zajete z mobilnimi napravami, vendar rezultati še vedno niso bili najboljši (tabela 3.2).

Ime Polja	Način zajema	Vseh slik	Pravilnih	%
Koda namena	Mobilna naprava	180	166	92,22
	Spletna kamera	100	98	98,00
Namen plačila	Mobilna naprava	180	142	78,89
	Spletna kamera	100	96	96,00
Znesek	Mobilna naprava	180	158	87,78
	Spletna kamera	100	99	99,00
IBAN	Mobilna naprava	180	149	82,78
	Spletna kamera	100	98	98,00
Referenca	Mobilna naprava	180	152	84,44
	Spletna kamera	100	98	98,00

Tabela 3.2: Rezultati branja vsebine iz polj univerzalnega plačilnega naloga po optimizaciji.

Poglavje 4

Zaključek

Naš sistem optičnega branja univerzalnega plačilnega naloga veliko bolje prebere vsebino slik, zajetih s spletno kamero. Razlogi za to so v tem, da so vse slike zajete z isto napravo, isto ločljivostjo, predvsem pa so to slike, vse zajete pri enaki svetlobi. V nasprotju s tem so slike z mobilnimi napravami zajete pri zelo različnih osvetlitvah, zato je tudi predprocesiranje veliko težje.

Po drugi strani pa smo s testiranjem ostalih funkcionalnosti sistema ugotovili, da je razen pri branju petih polj slik zajetih z mobilnimi napravami, pri izpolnjevanju svojih ostalih ciljev sistem zelo dober. Iskanje pozicije naloga na sliki je skoraj 100-odstotna, ne glede na način zajema slike, prav tako tudi računanje pozicij polj. Sistem je odlično prebral tudi vsebino OCR-vrstice pri skoraj vseh slikah.

Glede na dobljene rezultate, bi načeloma lahko začeli z uporabo sistema, saj vrstica OCR vsebuje vse podatke, ki so potrebni za pravilno izpolnitev in delovanje aplikacije za delo s plačilnimi nalogi, vseeno pa smo se odločili, da še počakamo in poskušamo izboljšati branje omenjenih petih polj, saj nekateri univerzalni plačilni nalogi vrstice OCR nimajo izpolnjene.

4.1 Nadaljnje delo

Prvi korak našega nadaljnjega dela je seveda izboljšava predobdelave in branja polj Koda namena, Namen plačila, Znesek, IBAN in Referenca, ki jih sistem prebere slabše od naših pričakovanj. V teku so že raziskave za čim boljšo predobdelavo in pripravo slike na optično branje ter izboljšave samega optičnega branja. Izvajajo se tudi različne optimizacije sistema in implementacije idej, ki jih najdemo v raziskavah, ter testiranje, v kolikšni meri s temi spremembami izboljšamo naš sistem.

V drugem koraku nadaljnjega dela bi implementirali obveščanje komitenta o rezultatu. Sistem trenutno v primeru, da ni znal prebrati vsebine iz slike, uporabniku ne vrne nobene povratne informacije. Da bi bilo to mogoče, mora biti sistem dovolj pameten, da uspešno zazna napako.

V primeru, da do napake pride že pri iskanju kvadratkov (sistem najde manj kot dva oziroma več kot tri kvadratke) ali pri rotaciji je to zelo enostavno, saj je napaka takoj vidna. Uporabniku se v tem primeru pošlje sporočilo, da je pri branju prišlo do napake in naj ponovno zajame sliko.

Odkrivanje napačnega branja vsebina univerzalnega plačilnega naloga pa bo v sistemu potekalo po naslednjem postopku.

- Prvo polje na univerzalnem plačilnem nalogu je Koda namena, zato bo prvo preverjanje, preverjanje tega polja. Vsebina polja Koda namena se mora ujemati z eno izmed vrednosti v šifrantu kod namenov plačil [21].

- V drugem polju, Namen plačila, vsebina ni z ničemer določena, zato se pravilnosti branja ne da preverjati. Sistem bo za to polje v vsakem primeru vrnil, kar bo prebral.

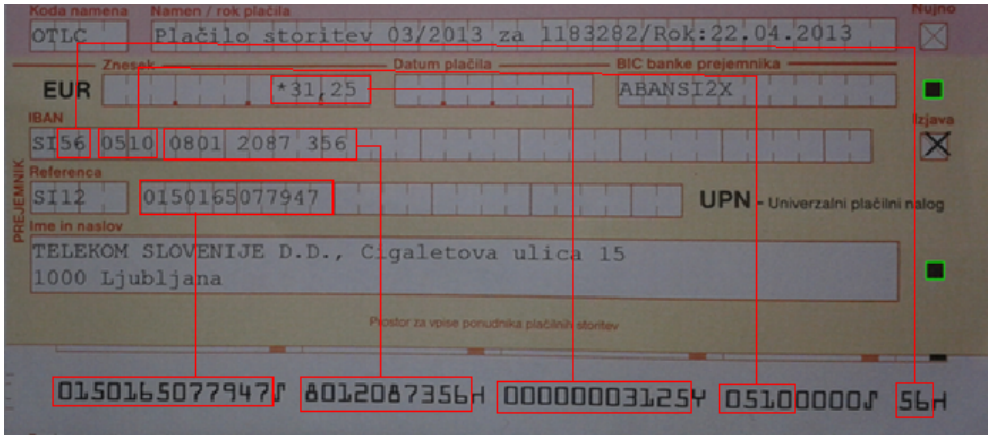
- Naslednje polje je polje Znesek. Tudi vsebina v znesku je lahko poljubno veliko število in ga ne moremo primerjati z neko drugo vrednostju. Za to polje bomo v sistemu preverjali, ali so vsi znaki števila in pozicijo pike in vejice, ter glede na to določali pravilnost.

- IBAN je v Sloveniji sestavljen iz 19 črkovnih in numeričnih znakov, kjer prvi 4 znaki predstavljajo oznako države (za Slovenijo je to vedno SI56), naslednjih 5 znakov predstavlja oznako banke in njene organizacijske enote, naslednjih 8 znakov predstavlja številko računa, zadnja 2 znaka pa sta kontrolni številki.

  Vsebino je sistem torej pravilno prebral takrat, ko so na prvih štirih mestih znaki SI56 in je ostanek pri deljenju števila, ki ga sestavlja ostalih 15 znakov, s številom 97 enak 1 [23].

- Vsebina polja Referenca je sestavljena iz konstante (SI ali RF), številke modela (2 številki) in vsebine modela (22 znakov, od tega 20 številk in največ dva vezaja). Prva dva znaka našega polja morata torej biti SI ali RF, pravilnost ostale vsebine pa preverimo s kontrolno številko, ki jo izračunamo, iz te vsebine, po postopkih, opisanih v [22].

- V zadnjem polju, vrstici OCR, imamo podatke reference, ibana in zneska, zato bo nam bo to služilo kot dodatna kontrola, za dodatno preverjanje pravilnosti prebrane vsebine (Slika 4.1).



Slika 4.1: Povezava med podatki v vrstici OCR in ostalimi polji.

Sistem bo v primeru odkritja napačno prebrane vsebine enega izmed polj uporabnika ustrezno obvestil o napaki.

Naslednji korak nadaljnjega dela bi bila izdelava mobilne aplikacije, ki bi zajela sliko in jo posredovala našemu sistemu v obdelavo in uveljavitev sistema na blagajnah Banke Koper.

Kot zadnji korak našega nadaljnjega dela, pa bi sistem nadgradili z inteligentnim optičnim branjem vsebine oziroma optičnim branjem ročno napisanih univerzalnih plačilnih nalogov.

Literatura

[1] Združenje bank Slovenije. Dostopno na: http://www.zbs-giz.si/zdruzenje-bank.asp?StructureId=933

[2] Predstavitev univerzalnega plačilnega naloga. Dostopno na: http://www.upn.si/

[3] Tehnični standard univerzalnega plačilnega naloga. Dostopno na: http://www.upn.si/uploads/public/tehnicni_standard_UPN.pdf

[4] Spletna stran Nove KBM. Dostopno na: http://www.nkbm.si/vsebina/10362/Nova-KBM-predstavila-najsodobnejso-mobilno-banko

[5] J.R.Parker, Pavol Federl, An Approach To Licence Plate Recognition. Dostopno na: http://pages.cpsc.ucalgary.ca/~federl/Publications/LicencePlate1996/licence-plate-1996.pdf

[6] Andolšek J., Kovačič S., Avtomatsko prepoznavanje registrskih tablic, seminar, Univerza v Ljubljani, 2003

[7] Image restoration. Dostopno na: http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/intro.html

[8] Leslie Stroebel, Richard D. Zakia, The Focal encyclopedia of photography, 1995

[9] RapidTables – RGB Color Codes Chart. Dostopno na: http://www.rapidtables.com/web/color/RGB_Color.htm

[10] TechTerms - Grayscale. Dostopno na:

http://www.techterms.com/definition/grayscale

[11] D. Ziou, S. Tabbone, Edge detection techniques: An overview, 1998

[12] J. Canny, A computational approach to edge detection, 1986

[13] Ding, Lijun and Goshtasby, Ardeshir, On the Canny Edge Detector, Pattern Recognition, 2001.

[14] G. V. Tcheslavski, Morphological Image Processing: Basic Concepts, 2009

[15] G. Bradski, A. Kaehler, Learning OpenCV - Computer Vision with the OpenCV Library, 2008

[16] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Transactions on Electronic Computers, 1961

[17] Vasile Prejmerean, Simona Motogona, Character recognition using morphological transformations. Dostopno na: http://www.cs.ubbcluj.ro/~studia-i/2001-1/9-Prejmerean.pdf

[18] Schantz, Herbert F., The history of OCR, optical character recognition, 1982

[19] Optical Character recognition (OCR) – how it works. Dostopno na: http://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/

[20] Teach ITC - Optical Mark Reader. Dostopno na: http://www.teach-ict.com/gcse_new/computer%20systems/input_devices/miniweb/pg13.htm

[21] Šifrant kod namenov plačil. Dostopno na:

http://www.ujp.gov.si/DocDir/Obrazec%20UPN/Kode%20namena%20placila_sifrant.pdf

[22] Pravilnik o plačilnih navodilih, izpiskih o prometu in stanju, delnih izpisih ter obrestnih listih. Dostopno na: http://www.mf.gov.si/fileadmin/mf.gov.si/pageuploads/Direktorat_za_zakladni%C5%A1tvo/Enotni_zakladniski_sistem_v_republiki_Sloveniji/Predpisi/Seznam_pravnih_podlag_s_podrocja_placilnega_prometa/Pravilnik_pn_izpiski_s_prilogami.pdf

[23] Banka Slovenije. Dostopno na:

http://www.bsi.si/placilni-sistemi.asp?MapaId=1453