Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Sušin

Interaktivno branje padavinskih grafov

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Peter Peer

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Interaktivno branje padavinskih grafov

Interactive reading of rainfall graphs

Tematika naloge:

Na osnovi našega programa za avtomatsko branje padavinskih grafov smo zaznali potrebo po razširitvi programa z interaktivnostjo, ki bo izboljšala rezultate branja. Implementirajte to razširitev, skupaj z zasnovo grafičnega uporabniškega vmesnika, zamenjavo glavne knjižnice programa, prilagoditvijo delovanja na širši spekter grafov, ob tem pa pazite na dobro uporabniško izkušnjo. Dobljene rezultate tako kvalitativno kot kvantitativno primerjajte z rezultati avtomatskega branja.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Nejc Sušin sem avtor diplomskega dela z naslovom:

Interaktivno branje padavinskih grafov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. 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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. septembra 2015 Podpis avtorja:

Kazalo

Povzetek

V tem delu je predstavljen interaktivni program, ki omogoča digitalizacijo padavinskih grafov. Program sprejme sliko optično prebranega papirnatega traku z narisanim grafom, rezultat pa so intenzitete padavin v izbranem časovnem intervalu. Temelji na avtomatskem postopku detekcije, ki mu lahko pomagamo s prilagajanjem parametrov ali pa zaznano krivuljo ročno popravimo. Med glavnimi metodami za popravljanje so označevanje območja, ki ga želimo izpustiti iz obravnave, dodajanje točk inverzije ter ročno risanje krivulje. S pomočjo interaktivnega programa je bilo obdelanih 58 padavinskih grafov, rezultati pa primerjani z uradnimi podatki z Agencije Republike Slovenije za okolje (ARSO). Interaktivnost dokazano omogoča obdelavo širšega spektra padavinskih grafov ter izboljša rezultate avtomatskega postopka.

Ključne besede: padavinski graf, računalniški vid, digitalizacija.

Abstract

An interactive program for the digitalization of rainfall charts is presented. A curve is extracted from a scanned picture of a paper strip and the intensity of rainfall is read at a given time interval. The program allows us to modify the parameters and manually correct the results of an automatic rainfall chart reading algorithm, upon which the interactive system is based. The main methods used to correct the readings are selecting an area to be excluded from further analysis, adding inversion points and manually drawing the curve. Using this interactive program, 58 rainfall charts were processed and the results were compared to those of the Environmental Agency of the Republic of Slovenia. We proved that the interactivity allows a wider spectrum of charts to be processed and gives more accurate results than the automatic procedure.

Keywords: rainfall chart, computer vision, digitalization.

Poglavje 1

Uvod

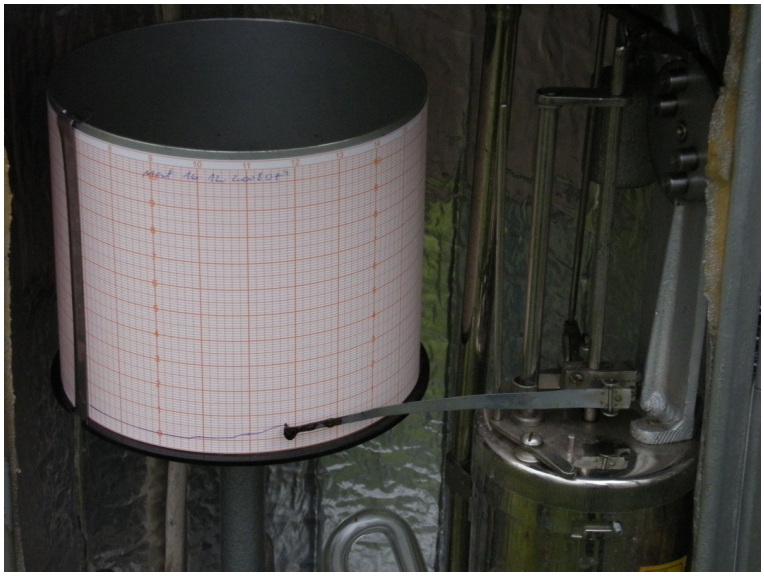
Geološki terminološki slovar definira padavine kot atmosfersko vodo, ki pade na zemljino površje [7].

Količina padavin je običajno izražena v milimetrih in predstavlja višino vodnega stolpca, ki se akumulira na površini enega kvadratnega metra. Za merjenje uporabljamo pluviometre in pluviografe. Pluviometer je lahko vsaka cilindrična posoda z izraženo skalo, iz katere ročno odčitamo količino zapadlih padavin v nekem časovnem intervalu (slika 1.1). Beleženje spreminjanja količine padavin skozi čas pa nam omogočajo avtomatski merilniki – pluviografi (slika 1.2). Pri teh se padavine nabirajo v zbiralni posodi, količina vode pa se sproti shranjuje – v obliki grafa na papir ali pa digitalno.

Primer na papir izrisanega grafa je viden na sliki 1.3. Na ordinatni osi je prikazan čas (24 ur), na abscisi pa količina vode, ki se je nabrala v merilni posodi (od 0 do 10 mm). Ko je posoda polna, se izprazni. Na grafu to opazimo kot nenaden spust krivulje do vrednosti 0 mm. Kadar padavin ni, se na graf izrisuje ravna horizontalna črta. Kadar so padavine močne, graf strmo narašča.

Za arhiviranje in preglednost papirnati trakovi niso najbolj primerni. Jasno je, da želimo te podatke za nadaljnjo analizo pretvoriti v računalniku prijaznejšo obliko. Ročni postopek digitalizacije zahteva veliko časa ter zbranosti. Računalniški program, ki bi ta postopek izvedel samodejno, bi proces

Slika 1.1: Preprost pluviometer.



Slika 1.2: Notranjost pluviografa, ki količino padavin izrisuje na papirni trak.

Slika 1.3: Primer padavinskega grafa.

bistveno pospešil oziroma poenostavil.

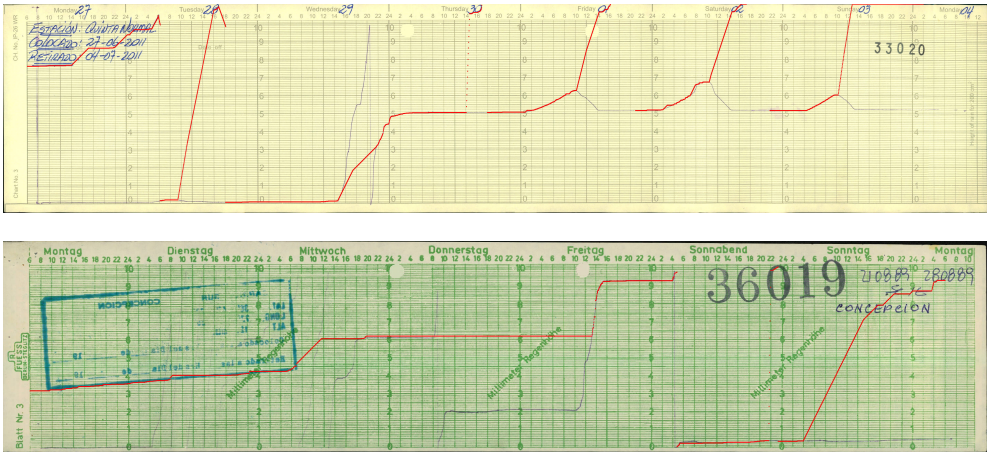
1.1 Sorodna rešitev

Papirnate padavinske grafe si želimo digitalizirati. Ta problem je osnova za diplomsko delo Računalniško branje padavinskih grafov [1]. Raziskava je bila predstavljena tudi v reviji Meteorological Applications [4]. Definira postopek, ki omogoča avtomatsko računalniško branje grafov pluviografskih trakov. Cilj dela je bil program, ki bo hitro in čim natančneje ter zanesljivo (tudi v primeru slabo zaznanega signala) vrnil rezultat. Predlagan postopek je bil implementiran in preizkušen na 58 grafih, rezultati pa primerjani s tistimi, ki jih hrani ARSO.

Implementiran program je deloval povsem samodejno. Preko ukazne vrstice je sprejel vhodno datoteko slike grafa ter željen interval pri izpisu rezultatov. Potek samega postopka je podrobneje opisan v poglavju 2.1.

1.2 Motivacija in cilji

Pobuda za nadaljnje delo je prišla s strani univerze v Čilu. Njihovi padavinski grafi se precej razlikujejo od slovenskih, program pa v tem pogledu ni bil prilagodljiv. Poleg tega je krivulja na nekaterih grafih prešibka za avtomatsko analizo in bi bili nujno potrebni ročni popravki. Dva taka primera sta prikazana na sliki 1.4.



Slika 1.4: Primera napačno zaznanih padavinskih grafov iz Čila.

Cilj naloge je nov oziroma nadgrajen program, ki temelji na obstoječem algoritmu, vključuje pa naslednje izboljšave:

- posodobitev z novimi knjižnicami
- nadgraditev z grafičnim uporabniškim vmesnikom
- dodatek interaktivnosti – možnost ročnih popravkov
- prilagoditev za uporabo na širšem spektru grafov
- objava izvorne kode in odprtokodna licenca.

1.3 Licenca

Izvorna koda programa je objavljena na spletu, na repozitoriju GitHub [9]. Zaščitena je pod GNU LGPL v3 licenco [10]. Naš cilj je, da se rešitev še naprej razvija in nadgrajuje, v skladu s potrebami uporabnikov. Odprtokodna licenca omogoča posameznikom, da program prilagodijo svojim potrebam. Omogočeno je tudi podajanje predlogov za izboljšave ter dodatne funkcionalnosti. Tako lahko tudi raziskovalci brez programerskega znanja (ki so, konec koncev, ciljni uporabniki programa) prispevajo k projektu. Na spletnem

repozitoriju se poleg izvorne kode nahajajo kratka navodila za razvijalce in delujoč program. Trenutno je podprt le operacijski sistem Windows, vendar prenosljivost kode ne bi smela biti problematična.

1.4 Struktura naloge

Jedro naloge je razdeljeno na tri dele. V drugem poglavju je predstavljen program za interaktivno branje padavinskih grafov – najprej obstoječ avtomatski postopek, za njim pa še nadgrajen, interaktivni postopek z grafičnim uporabniškim vmesnikom. Vključena sta tudi primera uporabe za dva različna padavinska grafa. Tretje poglavje je namenjeno implementaciji rešitve, od uporabljenih knjižnic ter predstavitev posameznih razredov do glavnih izzivov, na katere smo naleteli ob razvoju. V četrtem poglavju so podani kvalitativni in kvantitativni rezultati z diskusijo. Peto poglavje podaja zaključke naloge.

Poglavje 2

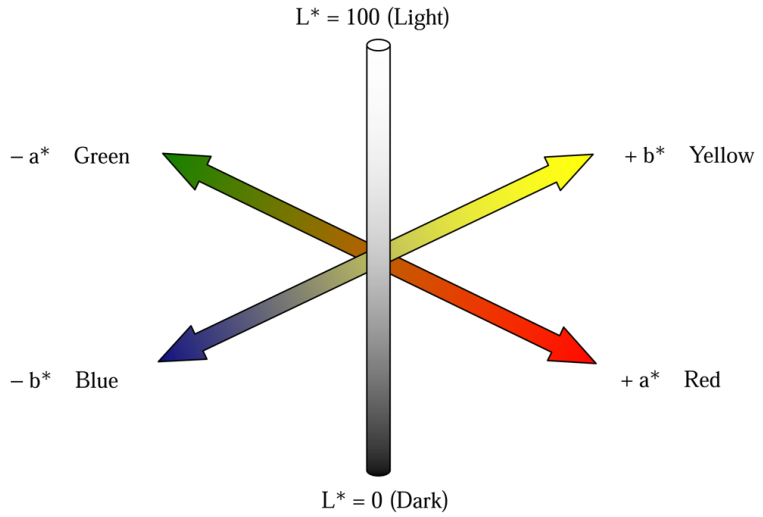
Program za interaktivno branje padavinskih grafov

V prvem delu poglavja je predstavljen algoritem za avtomatsko branje, ki je v veliki meri ostal nespremenjen [1, 4]. Posodobili smo knjižnice in izkoristili njihove nove funkcionalnosti, kodi smo dodali komentarje za lažje razumevanje, odpravili nekaj napak v programu in postopek optimizirali. Nadgradnje postopka za potrebe interakcije so opisane v drugem delu. Sledi še primer interaktivne obdelave za dva padavinska grafa.

2.1 Avtomatski postopek

Sliko padavinskega grafa obdelamo v štirih korakih, na koncu pa prikažemo rezultate. Postopek je prikazan na diagramu na sliki 2.1.

Slika 2.1: Koraki algoritma za avtomatsko branje padavinskih grafov.

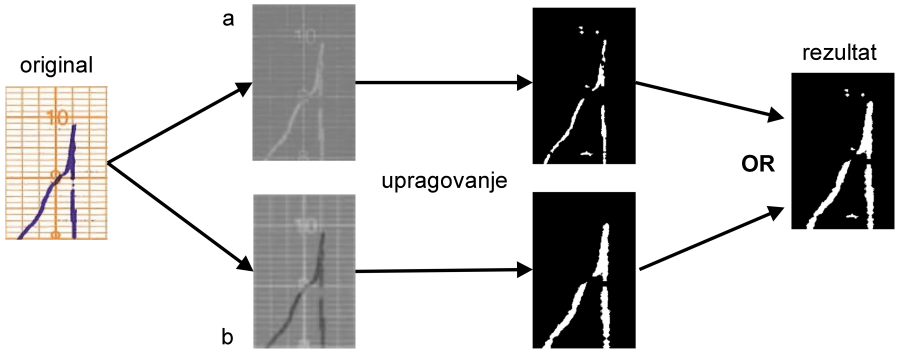


Slika 2.2: Barvo v prostoru CIELab predstavimo s tremi števili. Vsaka zavzame vrednost med 0 in 255.

2.1.1 Segmentacija

V prvem koraku želimo ločiti krivuljo od ozadja. Najprej barvni prostor slike pretvorimo iz RGB v CIELab, saj je ta za naše potrebe bolj primeren. CIELab je nelinearen prostor barv, čigar namen je čim bolje oponašati človeško oko. Če je razlika med dvema barvama v koordinatnem sistemu CIELab pod določenim pragom, bosta ti barvi za človeka enaki. Barvni prostor je sestavljen iz treh komponent, vsaka zavzame vrednost med 0 in 255: L (svetlost), a (zeleno ali rdeče) in b (modro ali rumeno) (slika 2.2).

Ko si ogledamo posamezne komponente slike v tem barvnem prostoru, vidimo, da krivulja (če ni prešibka) lepo izstopi od ozadja. Vzamemo komponenti a in b ter nad njima poženemo upragovanje z rastjo regij. Ta algoritem na sivinski sliki poišče vse točke, ki presegajo nek prag (v našem primeru naj bi ta prag presegale le točke, ki so del krivulje). To so začetne točke ali semena regij. Iz vsakega od teh semen potem nadaljuje s širjenjem regije, pri čemer kot kriterij za pripadnost regiji uporablja spodnji prag ter odstopanje sosednjih točk od povprečja regije. Rezultat segmentacije je unija obeh



Slika 2.3: Sliko v prostoru barv CIELab razdelimo na komponente. Nad komponentama a in b izvedemo upragovanje. Rezultat je unija dobljenih binarnih slik.

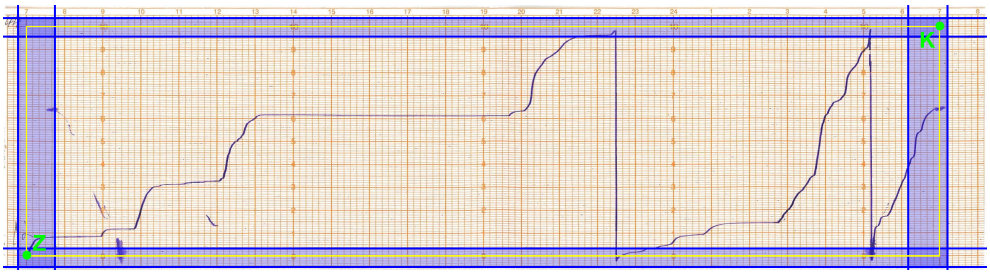
komponent po upragovanju. Slika 2.3 ilustrira celoten korak segmentacije.

2.1.2 Kalibracija

Poleg krivulje potrebujemo še podatek o ozadju grafa. Natančneje nas zanimajo skala, začetek in konec krivulje ter skrajne vrednosti. Vse to je zajeto v področju zanimanja, ki ga definiramo s točkama Z in K, označenima na sliki 2.4. Točka Z označuje čas, ko se meritev začne in y vrednost 0 mm. Točka K pa po x osi označuje čas, ko se meritev konča in največjo možno vrednost y (10 mm).

Meje področja zanimanja iščemo v vnaprej določenih okvirjih, ki so prav tako označeni na sliki 2.4. V vsakem od okvirjev se nad komponento b barvnega prostora CIELab izvede običajno upragovanje – zavržemo elemente, ki imajo vrednost nižjo od 145. V levem in desnem okvirju poiščemo stolpec z največjim številom preostalih elementov, v zgornjem in spodnjem pa namesto stolpca iščemo vrstico. Tako dobimo vse štiri meje področja zanimanja.

Če poznamo časovni razpon grafa in krivuljo poravnamo s področjem zanimanja imamo vse potrebno za izračun intenzitete padavin v posameznih časovnih intervalih.



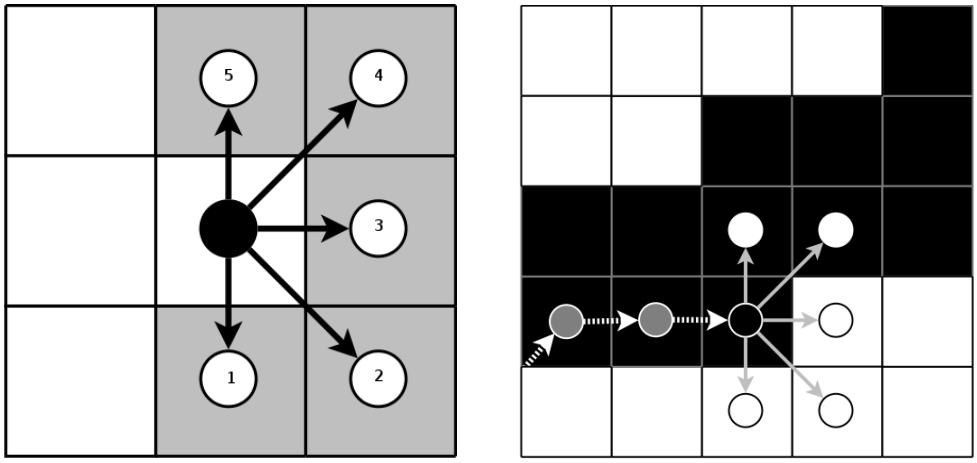
Slika 2.4: Področje zanimanja lahko definiramo kot kvadrat s spodnjim levim ogliščem Z in zgornjim desnim ogliščem K. Iščemo ga v štirih okvirjih, ki so obarvani modro. Z rumeno je označeno končno področje zanimanja.

2.1.3 Sledenje krivulji

Segmentacija nam v idealnem primeru vrne binarno sliko, na kateri imajo vsi slikovni elementi, ki so del krivulje, vrednost 255, ostali pa imajo vrednost 0. V vsakem stolpcu slikovnih elementov znotraj področja zanimanja želimo izbrati natanko eno točko, ki je del krivulje. Ker pa je debelina črte večja od enega slikovnega elementa, je v vsakem stolpcu v resnici več točk, ki pripadajo krivulji, poleg teh pa so običajno prisotni tudi madeži in nepravilnosti. Pri določanju krivulje se zato izmenjujeta dva postopka:

- *Lokalno sledenje spodnjega roba krivulje*, ki lahko hitro določi točen potek, dokler so točke povezane. Algoritem izbira točke v nasprotni smeri urinega kazalca, prične spodaj in konča zgoraj. Vrstni red obiskovanja in primer vidimo na sliki 2.5.

- *Globalno dvostopenjsko drseče povprečenje*, ki računa povprečje preko več sosednjih stolpcev. Nepravilnosti nimajo tako velikega vpliva, vendar je rezultat približek in ne natančen potek krivulje. Deluje v dveh korakih. Najprej izračuna drseče povprečje (angl. moving average) večih sosednjih stolpcev. To vrednost v drugem koraku uporabi kot središče manjšega okna, v katerem išče detektirano krivuljo.

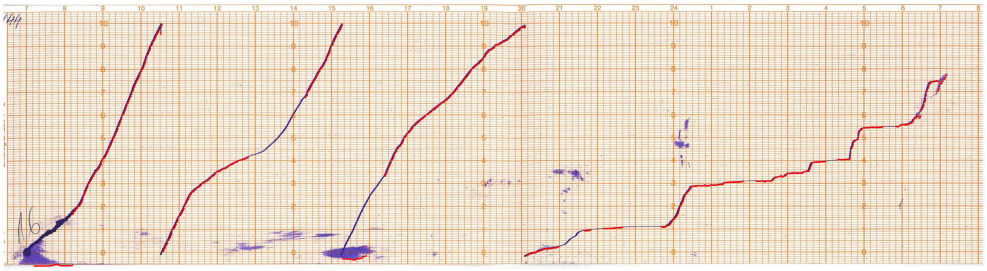


(a) Vrstni red obiskovanja.

(b) Konkreten primer sledenja.

Slika 2.5: Izbira naslednjega koraka pri sledenju roba.

Dokler je to mogoče, se izvaja sledenje roba. Če algoritem ne najde poti naprej, preklopi na drseče povprečenje. Ko oceni, da je rezultat povprečenja dovolj natančen, se vrne na sledenje roba. Tako lahko sorazmerno natančno določimo potek krivulje. Kot vidimo na sliki 2.6, so lahko še vedno prisotne nepravilnosti, zato je potrebno rezultat še izpopolniti.



Slika 2.6: Včasih je sledenje krivulji manj uspešno – prisotne so luknje v zaporedju, madeži so zaznani kot del krivulje.

2.1.4 Naknadna obdelava

Zadnji korak obdelave služi odpravljanju napak in glajenju krivulje. Sedaj ne obravnavamo več slike, temveč seznam točk, ki smo ga dobili v prejšnjem koraku. Naknadno obdelavo sestavljajo naslednje funkcije:

- *Zavračanje točk madežev* odstrani madeže, ki bi lahko zmotili iskanje inverzij. Prepoznamo jih po nenadnih skokih zaporedja, ki jim sledi skok v nasprotno smer.

- *Iskanje točk inverzije* poišče mesta, kjer krivulja nenadoma pade, kar pomeni, da se je posoda izpraznila. To so mesta, kjer se krivulja iz zgornje tretjine področja zanimanja nenadoma spusti do vrednosti 0. Te točke shranimo, saj ločujejo krivuljo na več odsekov.

- *Omejitev na naraščajoče funkcije* poskrbi, da znotraj vsakega od odsekov, ki smo jih določili s točkami inverzije, krivulja strogo narašča. Z algoritmom za iskanje najdaljšega naraščajočega podzaporedja aproksimiramo pravilen potek krivulje in tako izločimo morebitne nepravilnosti.

- *Zapolnitev manjkajočih vrednosti* z linearno interpolacijo zapolni presledke, ki so prisotni bodisi že od prejšnjega koraka, bodisi smo jih pridelali v koraku naknadne obdelave.

2.1.5 Izpis rezultatov

Rezultat obdelave je seznam točk, kjer vsaka točka predstavlja višino vode, akumulirane v zbiralni posodi do tistega trenutka. Število točk je enako širini področja zanimanja – manjkajočih vrednosti ni, saj smo jih eliminirali v zadnjem koraku naknadne obdelave. Želimo izpis intenzitete padavin v izbranem časovnem intervalu. Ta je enaka razliki v višini krivulje med dvema sosednjima intervaloma. Primer izpisa je na sliki 2.7.

```
2006-03-04  22  0.1
2006-03-04  23  0.3
2006-03-04  24  0.2
2006-03-05  1  0.0
2006-03-05  2  0.4
2006-03-05  3  0.3
```

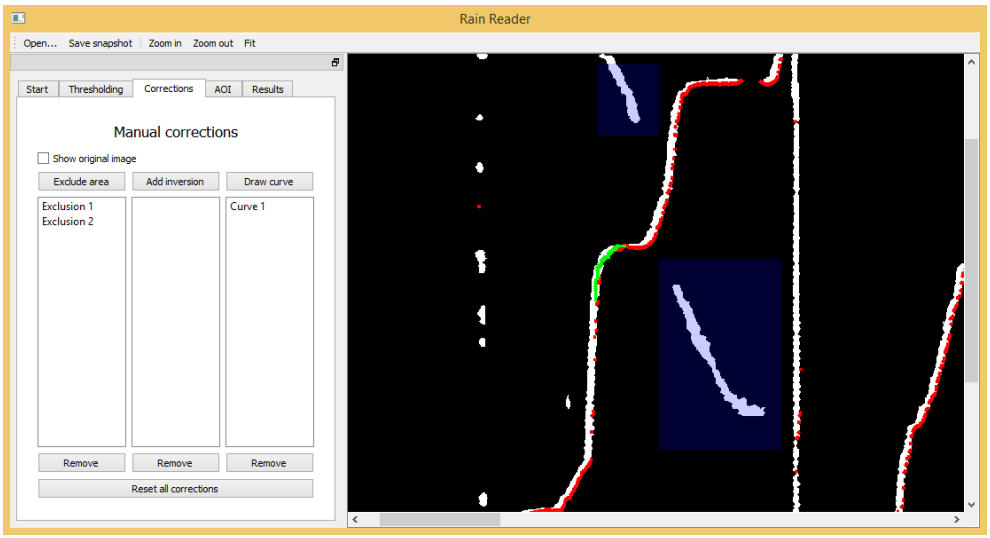
Slika 2.7: Del izpisa avtomatske različice programa na urnem intervalu. Vidimo, da je 4. marca 2006 med 22. in 23. uro zapadlo 0,3 mm padavin.

2.2 Interaktivni postopek

Ob zagonu programa uporabnika pričaka začetno okno s kratkim opisom ter osnovnimi podatki programa. Odpiranje datoteke je edina v tem koraku izvedljiva operacija. Program zna brati slike formata JPEG in PNG. Ob naložitvi nove slike se najprej izvede samodejni postopek branja grafa in rešitev je ponujena uporabniku. Ta presodi, ali je bil avtomatski postopek uspešen – v tem primeru lahko shrani rezultate in zaključi z delom, oziroma nadaljuje delo z odpiranjem naslednje slike. Če je pri prepoznavanju prišlo do napak, lahko uporabnik poljubno preklaplja med štirimi fazami obdelave, dokler rezultat ni zadovoljiv. Grafični vmesnik med uporabo vidimo na sliki 2.8. Posamezne faze so opisane v nadaljevanju, grafični vmesnik za vsako pa je prikazan na sliki 2.9.

2.2.1 Segmentacija

Uporabniški vmesnik za korak segmentacije je na sliki 2.9a. Uporabnik si lahko ogleda sliko pred segmentacijo ter po njej, možen je ogled posamezne barvne komponente ali celotne slike. S pomočjo vrtilnih polj lahko prilagaja parametre za postopek upragovanja z rastjo regij za vsako barvno komponento posebej. S klikom na gumb Apply potrdi spremembe in posodobi sliko.



Slika 2.8: Program med uporabo.

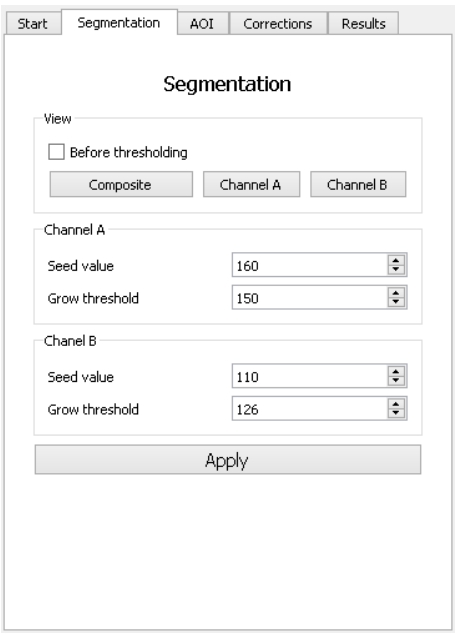
2.2.2 Področje zanimanja

Uporabniški vmesnik za ta korak je na sliki 2.9b. Z uporabo štirih vrtilnih polj premikamo meje področja zanimanja. Vsaka sprememba je nemudoma tudi vidna na sliki. Vrednost lahko spreminjamo s klikom na kontrolo, z vrtljaji miškinega koleščka, s puščicami na tipkovnici ali z ročnim vnašanjem vrednosti.

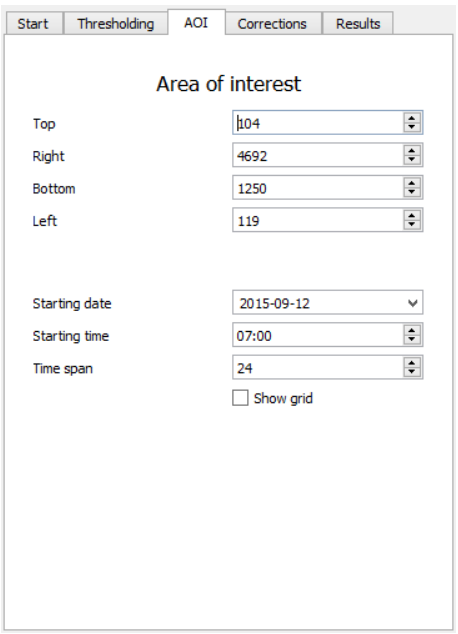
V tem koraku tudi določimo datum in čas, ko se je merjenje začelo. Na podlagi tega datuma se bodo izpisali rezultati. Določimo še razpon, ki ga ima graf – časovno obdobje, ki ga graf pokriva, v urah. Privzeta vrednost je 24 ur. Da smo področje zanimanja pravilno nastavili, lahko preverimo tako, da vklopimo mrežo. Ta na podlagi časovnega razpona izračuna pozicije vertikalnih črt in jih prikaže. Ta mreža se mora pokriti z mrežo grafa.

2.2.3 Ročni popravki

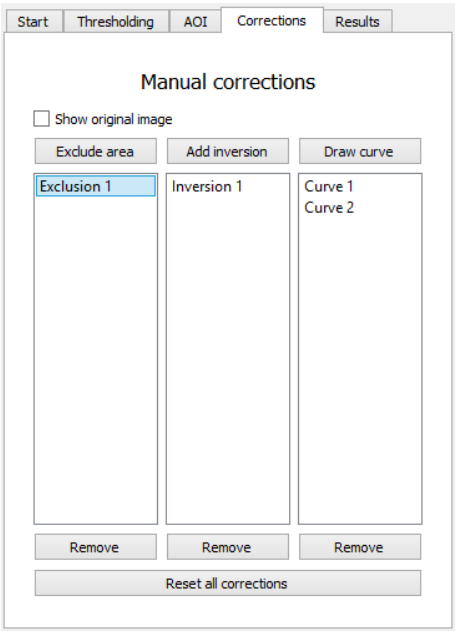
Uporabniški vmesnik za ročne popravke je na sliki 2.9c. Uporabnik lahko v tem koraku izvaja ročne popravke napak, ki jih opazi med rezultati. Korak se nahaja med sledenjem krivulje in naknadno obdelavo, kar pomeni, da so



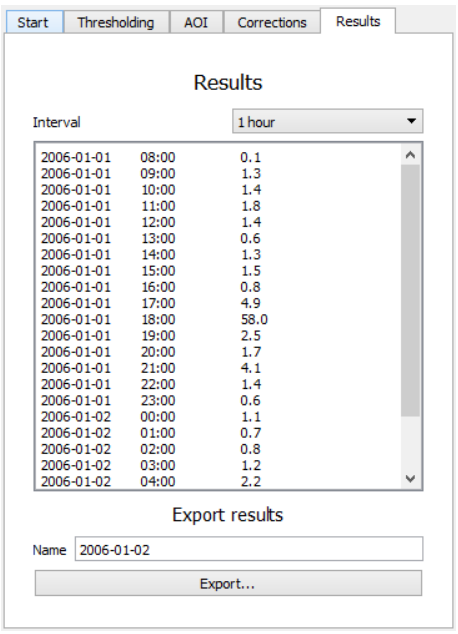
(a) Segmentacija



(b) področje zanimanja

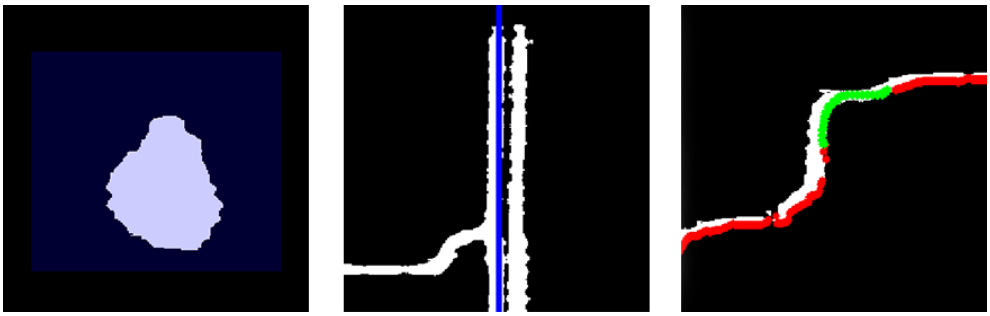


(c) ročni popravki



(d) rezultati

Slika 2.9: Grafični vmesnik programa. Zavihek `Start` (ni prikazan na sliki) vsebuje kratka navodila in osnovne podatke programa.



Slika 2.10: Primer neupoštevanja območja, dodane točke inverzije in ročno narisanega dela krivulje (zeleno).

na sliki prisotni madeži ter nepravilnosti, ki so kasneje v koraku naknadne obdelave samodejno odpravljene. Uporabnik naj ročne popravke izvaja le tam, kjer je to potrebno. Možni popravki so:

- *Neupoštevanje območja:* označeno polje ne vsebuje relevantnih informacij za procesiranje, zato naj ga program izpusti iz nadaljnje obravnave. Na ta način izločimo madeže, ki jih je avtomatski postopek napačno zaznal.
- *Točke inverzije:* kadar so točke inverzije preblizu skupaj ali premalo jasne, jih algoritem ne prepozna samodejno. S tem orodjem povlečemo vertikalo na mesto, kjer pride do inverzije. Ponovno je potrebna previdnost uporabnika, da ne označuje inverzij, ki jih algoritem že samodejno zazna.
- *Risanje krivulje:* S pomočjo miške lahko uporabnik nariše del krivulje, ki v samodejnem postopku ni bila pravilno zaznana. Popolna oblika ni toliko pomembna, saj gredo tudi točke, ki jih nariše uporabnik, čez postopek glajenja, da so linije čiste.

Vsaka vrsta popravkov ima svoj seznam, v katerega se dodajajo vnosi. Če uporabnik katerega označi, se ta poudari na sliki. Klik na gumb `Remove`, odstrani izbrani popravek. Enako funkcijo ima tipka `delete` na tipkovnici. Vse tri vrste popravkov so prikazane na sliki 2.10.

2.2.4 Rezultati

Uporabniški vmesnik za prikaz rezultatov je na sliki 2.9d. Ko se prestavimo v korak rezultatov, se sproži naknadna obdelava in na sliki se izriše končni rezultat poteka krivulje. Rezultate lahko izpisujemo v različnih časovnih intervalih, ki jih izberemo v kombiniranem seznamu. Določimo lahko tudi geografsko lokacijo oziroma ime grafa, ki bo služilo za poimenovanje izvožene datoteke. Primer izpisa rezultata na urnem intervalu je viden na sliki 2.11.

Uporabnik lahko med temi koraki prehaja poljubno. Vsaka sprememba se sproti shrani in bo vidna v naslednjem koraku. Ves čas se lahko uporabnik sprehaja po prikazani sliki s pomočjo miške, gumbi v orodni vrstici omogočajo približevanje in oddaljevanje pogleda. Gumb `Save snapshot` shrani sliko grafa v polni ločljivosti, v načinu, ki je trenutno prikazan. Načini prikazovanja slike so:

- izvorna slika
- izvorna slika z označenim področjem zanimanja
- različni koraki segmentacije (posamezne komponente CIELab pred in po segmentaciji, končni produkt)
- izvorna ali segmentirana slika z označenimi ročnimi popravki
- končni rezultat.

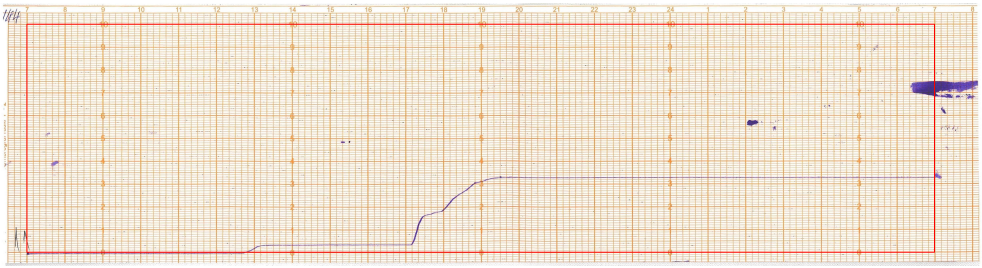
2.3 Primer postopka

Oglejmo si postopek obdelave na dveh različnih grafih.

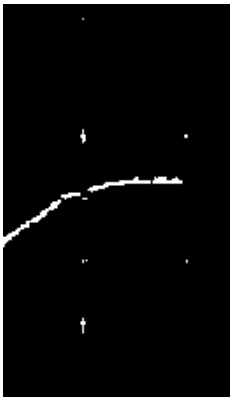
Ker je bil avtomatski postopek razvit posebej za slovenske grafe, jih prebere zelo uspešno. Slika 2.12 prikazuje primer obdelave slovenskega grafa. Področje zanimanja je pravilno določeno, meje za upragovanje so postavljene dobro, vendar jih lahko še izboljšamo. Ročni popravki so potrebni samo za odpravljanje madežev ob koncu krivulje.

```
1  2006-02-26  08:00  0.1
2  2006-02-26  09:00  0.0
3  2006-02-26  10:00  0.0
4  2006-02-26  11:00  0.0
5  2006-02-26  12:00  0.0
6  2006-02-26  13:00  0.0
7  2006-02-26  14:00  0.0
8  2006-02-26  15:00  0.0
9  2006-02-26  16:00  0.0
10 2006-02-26  17:00  0.3
11 2006-02-26  18:00  0.1
12 2006-02-26  19:00  0.1
13 2006-02-26  20:00  0.1
14 2006-02-26  21:00  0.1
15 2006-02-26  22:00  0.6
16 2006-02-26  23:00  0.8
17 2006-02-27  00:00  0.7
18 2006-02-27  01:00  0.7
19 2006-02-27  02:00  0.4
20 2006-02-27  03:00  0.2
21 2006-02-27  04:00  0.2
22 2006-02-27  05:00  0.1
23 2006-02-27  06:00  0.1
24 2006-02-27  07:00  0.1
```

Slika 2.11: Primer obdelave grafa, zajetega 26. in 27. februarja 2006. V prvi vrstici vidimo, da je 26. februarja med 7:00 in 8:00 zapadlo 0,1 mm dežja.

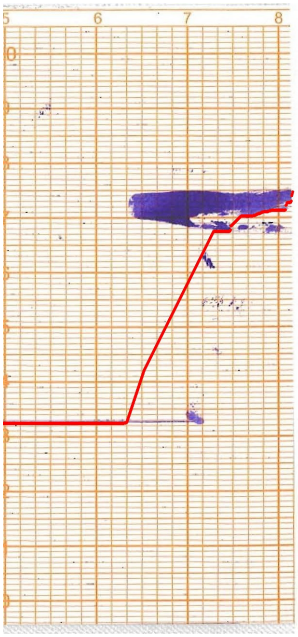


(a) Področje zanimanja je določeno pravilno.

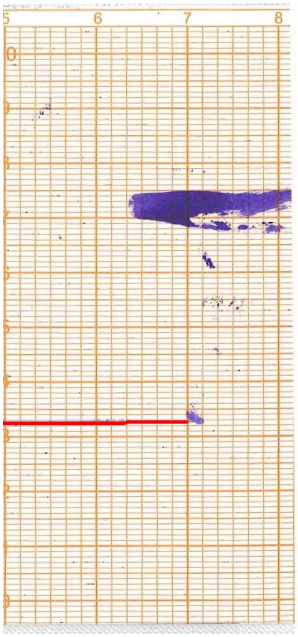


(b) Privzeto upragovanje.

(c) Popravljene meje upragovanja.



(d) Madež na koncu krivulje pokvari rezultat.

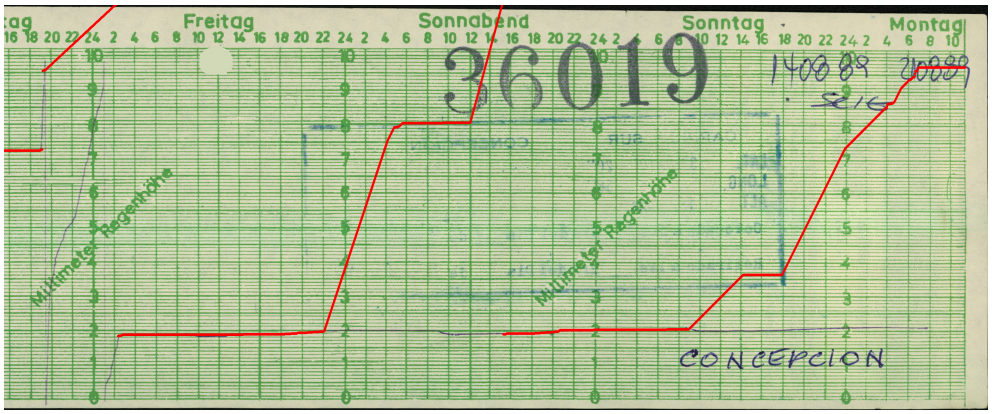


(e) Ročno smo odstranli madež in dobili pravilen rezultat.

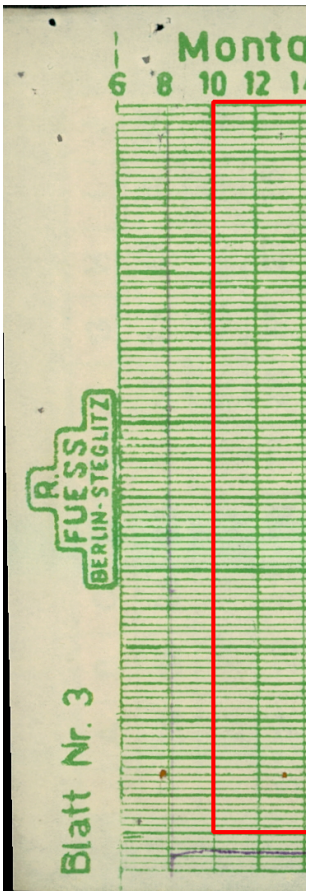
Slika 2.12: Primer obdelave slovenskega grafa.

Drugod po svetu je format padavinskih grafov lahko drugačen. Uporabljen je lahko drugačen papir, drugo črnilo, lahko so padavine zajete tudi skozi daljši čas. V tem primeru avtomatski postopek ni uspešen, saj ni prilagojen na drugačne pogoje. Iz Čila smo prejeli 9 zelo drugačnih grafov, žal brez dejanskih podatkov o padavinah.

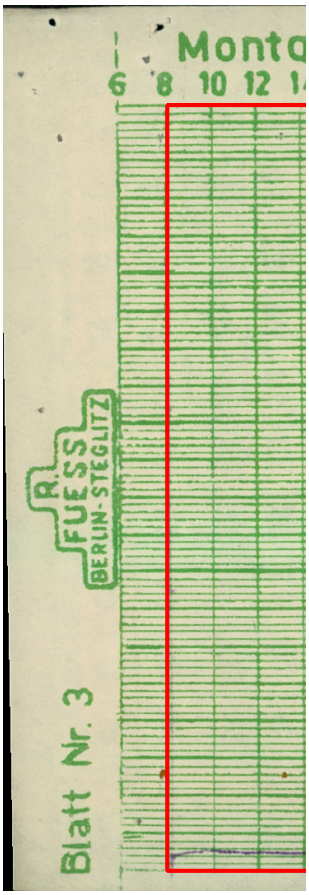
Slike 2.13, 2.14 in 2.15 prikazujejo primer obdelave grafa iz Čila. Potrebno je prilagoditi področje zanimanja na levem, spodnjem in desnem robu (slika 2.13b in 2.13c). Graf pokriva obdobje enega tedna, zato prilagodimo vrednost časovnega razpona. Prilagodimo tudi uro, saj se meritev začne ob 8:00. Upragovanje nam pri privzetih vrednostih ne daje koristnih rezultatov. Ustrezno prilagodimo meje, da je krivulja vidna (slika 2.14a in 2.14b). Preverimo končni rezultat, da vidimo, koliko popravkov bo potrebnih. Dodamo območje neupoštevanja po celotni širini grafa ob zgornjem in spodnjem robu, saj se je tam naredila bela obroba, ki je napačno zaznana kot del krivulje. Prav tako iz nadaljnje obdelave izključimo velik odtis štampiljke na desni (slika 2.14c in 2.14d). Po potrebi bi dele krivulje še ročno narisali ali dodali točko inverzije. Končni rezultat je pravilen potek krivulje (slika 2.15).



(a) Samodejno zaznan potek krivulje je popolnoma napačen. Prikazan je le del grafa.

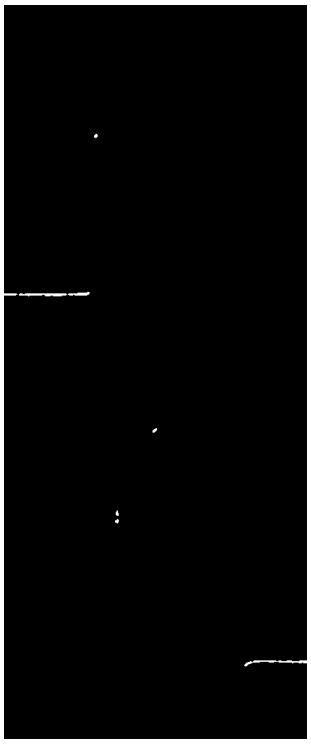


(b) Samodejno zaznano področje zanimanja.

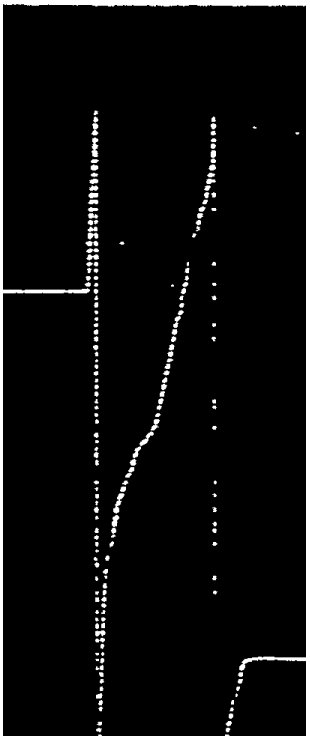


(c) Popravljeno področje zanimanja.

Slika 2.13: Primer obdelave grafa iz Čila.

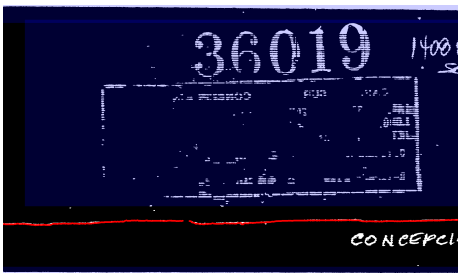


(a) Privzeto upragovanje.



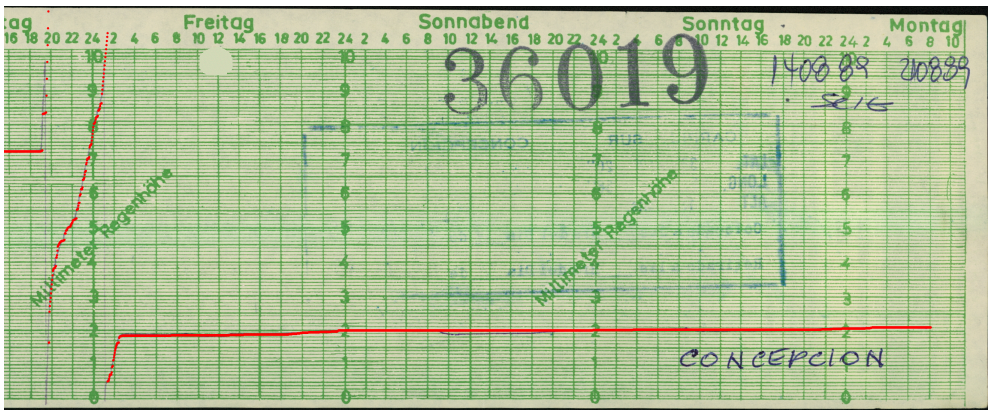
(b) Popravljene meje upragovanja.

(c) Krivulja pred ročnimi popravki takorekoč sploh ni zaznana.



(d) Krivulja po ročnih popravkih.

Slika 2.14: Primer obdelave grafa iz Čila.



(a) Končni rezultat interaktivne obdelave. Prikazan je le del grafa.

Slika 2.15: Primer obdelave grafa iz Čila.

Poglavje 3

Implementacija

V tem poglavju je predstavljena zgradba programa ter glavni izzivi pri njegovi implementaciji.

3.1 Knjižnice

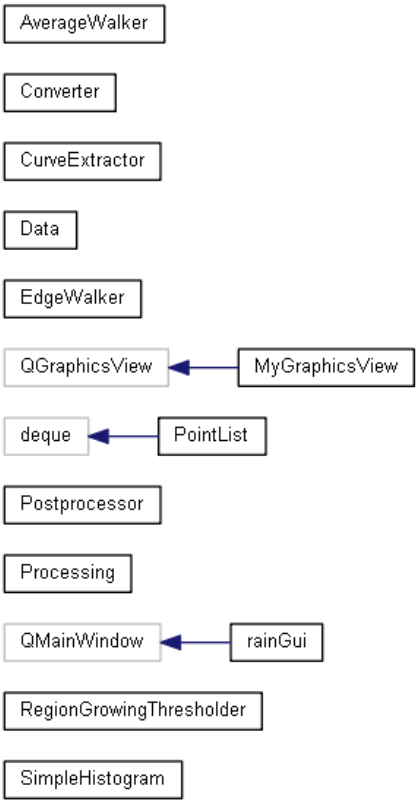
OpenCV je odprtokodna knjižnica, ki vsebuje preko 500 funkcij z različnih področij računalniškega vida. Mi smo jo uporabili za izvajanje operacij nad slikami grafov. Uporabili smo različico 3.0 [12].

Qt je odprtokodno ogrodje za izdelavo grafičnih uporabniških vmesnikov. Omogoča hitro izgradnjo naprednih uporabniških vmesnikov za različne platforme. Uporabili smo različico 5.5.0 [13].

Boost je velika zbirka splošno uporabnih C++ knjižnic. Poslužujemo se modula `datetime` [8] za formatiranje in računanje pravilnih datumov pri izpisu rezultatov. Trenutna različica knjižnice je 1.58.0.

3.2 Arhitektura rešitve

Program je sestavljen iz 13 izvornih C++ datotek, vsaka ima tudi pripadajočo zglavno datoteko (angl. header file). Sama struktura uporabniškega vmesnika je shranjena v ločenih datotekah, ki pa so generirane samodejno, s



Slika 3.1: Razredni diagram programa.

strani Qt okolja, zato jih ne bomo posebej izpostavljali. Razredni diagram je razviden s slike 3.1.

- `Main` služi le kot vstopna točka, ki požene grajenje vmesnika.
- `RainGui` pripravi uporabniški vmesnik in se odziva na uporabnikove ukaze tako, da kliče metode v razredu `Processing`, od katerega potem prejme slike ter jih prikaže.
- `Processing` je osrednji razred, ki sprejema ukaze za procesiranje, preda delo ustreznim razredom, hrani vmesne in končne rezultate ter posreduje le-te nazaj razredu za uporabniški vmesnik.

- `Pointlist` je podatkovna struktura, ki hrani dvostrani seznam (angl. deque) točk. Vanj shranimo rezultat sledenja krivulji. Dvostrani seznami nam omogočajo, da dodajamo in odstranjujemo elemente na začetku in koncu seznama, kar nam koristi pri naknadni obdelavi.

- `MyGraphicsView` je podrazred Qtjevega `QGraphicsView` za prikazovanje slik. Omogoča nam sledenje miškinega kazalca za risanje po sliki ter polaganje črt.

- `RegionGrowingThresholder` opravi upragovanje z rastjo regij.

- `Data` poišče, hrani ter omogoča spreminjanje podatkov o področju zanimanja.

- `CurveExtractor` je glavni razred za sledenje krivulji.

- `AverageWalker` izvede postopek za globalno dvo-stopenjsko drseče povprečje.

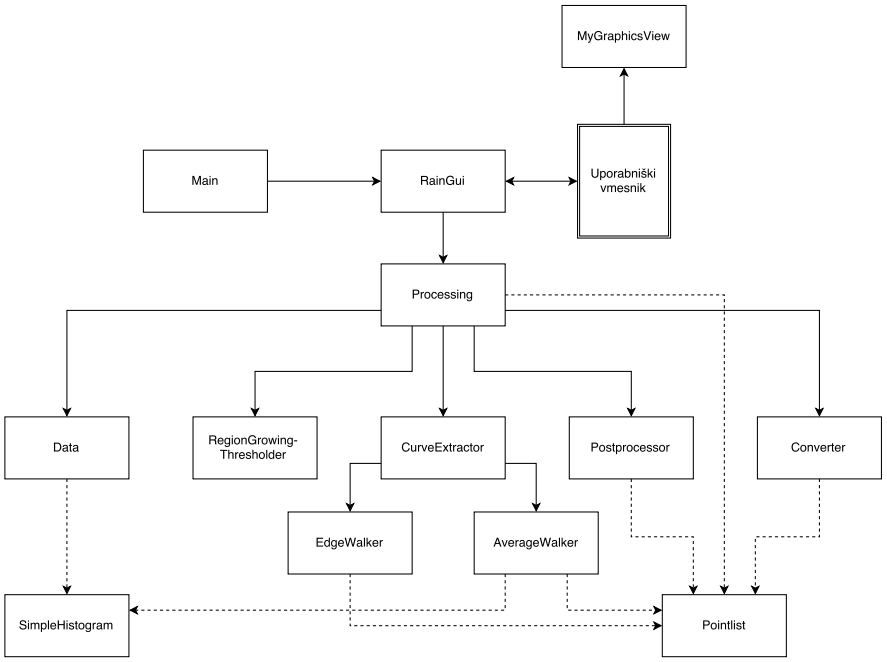
- `EdgeWalker` izvede postopek za sledenje spodnjega roba krivulje.

- `Postprocessor` izvede naknadno obdelavo zaznanih točk krivulje.

- `Converter` pretvori koordinate točk v podatke o padavinah ter sestavi seznam rezultatov na podlagi želenega intervala.

- `SimpleHistogram` je implementacija preprostega histograma, ki ga uporabimo pri iskanju področja zanimanja in zaznavanju krivulje.

Na sliki 3.2 so ilustrirane interakcije med razredi. Uporabniški vmesnik sporoča zahteve uporabnika in prejema odzive od razreda `RainGui`, ta pošilja signale razredu `Processing`, ki prelaga delo na ostale razrede ter zbira rezultate. Povezave s `SimpleHistogram` in `Pointlist` so črtkane, saj gre bolj za podatkovni strukturi kot razrede, ki opravljajo delo.



Slika 3.2: Diagram klicev med razredi. Entiteta `Uporabniški vmesnik` predstavlja skupek samodejno generiranih razredov.

```
1 // LTILib sprejme koordinate zgornje leve ter spodnje desne točke
2 ltilibRectangle = irectangle(x1, y1, x2, y2);
3
4 // OpenCV sprejme zgornjo levo točko ter širino in višino kvadrata
5 opencvRectangle = irectangle(x, y, width, height)
```

Slika 3.3: Konstruktorja za kvadrat v knjižnicah LTILib in OpenCV.

3.3 Glavni izzivi

3.3.1 LTI-Lib

Ne-interaktivna različica programa je operacije nad slikami izvajala s pomočjo knjižnice LTI-Lib [11]. Ta knjižnica se je do danes zelo spremenila, zato stara koda z novejšimi različicami ni združljiva. Podprta je na operacijskih sistemih Windows in Linux. Odločili smo se, da preidemo na OpenCV. Ta nudi več funkcionalnosti, je bolje dokumentirana in optimizirana za hitrost. Knjižnici imata podobno funkcionalnost in osnovne operacije so si zelo podobne. Seveda pa so prisotne majhne razlike, ki lahko povsem spremenijo delovanje. Potrebno je bilo preučiti obe knjižnici ter prilagoditi klice funkcij ter definicije osnovnih gradnikov. V nadaljevanju je opisanih nekaj razlik med knjižnicama.

Definicija kvadrata

Razreda, ki hranita podatke o kvadratu sta `lti::rectangle` ter `cv::Rect`. LTILib definira kvadrat s pomočjo njegove zgornje leve ter spodnje desne točke. OpenCV pa hrani kvadrat kot točko (zgornjo levo) ter njegovo širino in višino. Konstruktorja za oba razreda vidimo na sliki 3.3.

Upragovanje

Pri iskanju področja zanimanja se poslužimo navadnega upragovanja. LTI-Lib zahteva, da ročno določimo vse parametre upragovanja, medtem ko ima

```
1 thresholding::parameters thParams;
2 thParams.inRegionValue = 1.0;
3 thParams.outRegionValue = 0.0;
4 thParams.keepInRegion = true;
5 thParams.keepOutRegion = false;
6 thParams.lowThreshold = 145.0/255.0;
7 thresholding thresholder(thParams);
8 thresholder.apply(ch, image);
```

Slika 3.4: Primer sestavljanja argumentov za upragovanje v knjižnici LTILib.

```
1 cv::threshold(ch, binImg, 145.0, 255.0, cv::THRESH_TOZERO_INV);
```

Slika 3.5: Klic funkcije za upragovanje, vgrajene v OpenCV.

OpenCV različne načine delovanja že določene. Nam v tem primeru ustreza način `threshold to zero inverse`, pri katerem so elementi z vrednostjo nad podano mejo nastavljeni na 0, ostali pa ohranijo svojo vrednost. Sliki 3.4 in 3.5 prikazujeta izvedbo upragovanja v obeh knjižnicah.

Pretvorba v CIELab

LTILib ne podpira pretvorbe v barvni prostor CIELab ter delitve na njegove komponente. Ta postopek je bil zato ročno implementiran. OpenCV nam tu močno olajša delo, saj ima to funkcionalnost že vgrajeno. Želen rezultat dosežemo preprosto s klicem dveh funkcij, kot vidimo na sliki 3.6.

3.3.2 `MyGraphicsView`

Qt gradnik, ki prikazuje grafične elemente se imenuje `QGraphicsView`. Privzeto vključuje ta razred tudi način delovanja, kjer z miško na način povleci in spusti označujemo elemente. Ne vsebuje pa podpore za risanje, kjer bi miškin kazalec med držanjem gumba puščal sled. Še bolj specifična za naš program je možnost polaganja vertikalnih črt. Za učinkovito realizacijo omenjenih

```
1 cv::cvtColor(imCie, imCie, CV_BGR2Lab);
2 cv::split(imCie, channels);
```

Slika 3.6: Sliko `imCie` pretvorimo v barvni prostor CIELab ter jo razdelimo na posamezne komponente. Seznam le-teh se shrani v polje `channels`.

funkcionalnosti smo implementirali podrazred `MyGraphicsView`, ki omogoča tudi ta načina delovanja.

3.3.3 Uporabniška izkušnja

Postopek branja grafov ima pet glavnih korakov: segmentacija, kalibracija, zaznavanje, naknadna obdelava in prikaz rezultatov. V vsakem korakov se izvede nek značilen postopek, ki pripravi sliko na naslednji korak. Vsak od teh postopkov vključuje tudi parametre, ki so izbrani avtomatsko ali pa so vnaprej določeni. Namen interaktivnega programa je, da uporabniku omogoči spreminjanje teh parametrov po lastni presoji. Vsak korak ima s seboj povezano tudi povratno informacijo, ki mora biti vidna uporabniku, da ta ve, kako so spremembe vplivale na rezultat.

Smiselno je torej, da je postopek tudi v interaktivnem načinu razdeljen na korake, ki ustrezajo postopku v ozadju. V vsakem koraku so ponujeni parametri, ki jih lahko uporabnik prilagaja in relevantna povratna informacija, to je ustrezni način prikaza slike in obdelava le-te. Na ta način uporabnika vodimo skozi postopek in hkrati dajemo vpogled v samo delovanje algoritma, kar je dobro, saj bo z boljšim razumevanjem lahko delal hitreje in bolj učinkovito.

Pri načrtovanju uporabniškega vmesnika smo se opirali na deset Nielsenovih principov za načrtovanje uporabniških vmesnikov [6]:

- vidljivost sistema
- povezava z resničnim svetom
- uporabnikova svoboda

- konsistentnost in standardi
- preprečevanje napak
- prepoznaj raje, kot pomni
- fleksibilnost in učinkovitost
- minimalizem
- pomagaj uporabniku prepoznati in odpraviti napake
- pomoč in dokumentacija.

Vidljivost sistema pravi, naj uporabnik vedno vidi, kaj se dogaja s programom. Ko se v ozadju izvaja zahtevnejša operacija in mora uporabnik počakati, da se zaključi, mu to sporočimo z obvestilom Processing..., hkrati pa spremenimo miškin kazalec v način zasedenosti (angl. busy). Tako uporabnik ve, da je v teku operacija, obenem pa mu je onemogočeno klikanje na ostale kontrole. Zavihki nam zagotavljajo, da uporabnik vedno ve, v katerem koraku se nahaja.

Povezava z resničnim svetom se vidi pri izvajanju popravkov, kjer uporabnik riše neposredno na sliko. Tudi zavihki predstavljajo intuitiven način preklapljanja med stanji.

Ker popravki običajno niso potrebni na vseh korakih, je pomembno, da postopek ni preveč strogo voden, zato so koraki udejanjeni kot zavihki. Uporabnik lahko med njimi poljubno preklaplja – *uporabnikova svoboda.*

Gumbi, ki se pojavijo v programu imajo prepoznavne napise, ki so uporabniku poznani in jasno sporočajo namen: `Apply`, `Remove`, `Open...` , `Export...` . Postavitev vmesnika je *konsistentna in v skladu s standardi.*

Uporabniku želimo *omejiti možnosti napak.* Vsi gumbi, ki uporabniku trenutno niso koristni, so onemogočeni. Prav tako želimo razbremeniti uporabnikov spomin – *prepoznaj raje, kot pomni*, zato so vse za trenutni korak relevantne informacije vidne na ekranu.

Za pogoste operacije (približevanje, brisanje vnosov) so na voljo bližnjice na tipkovnici, ki pospešijo uporabo ter povečajo *fleksibilnost in učinkovitost*.

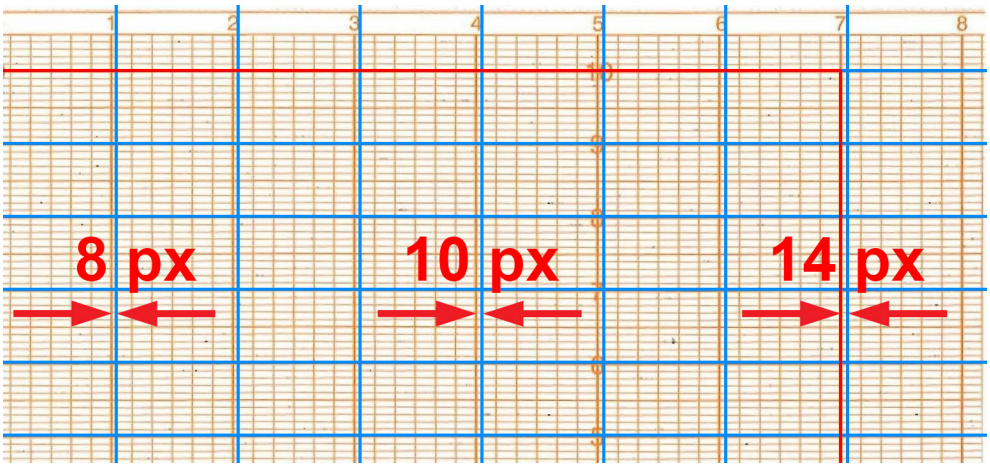
Odvečnih podatkov, ki bi upočasnjevale ali zmedle uporabnika, ni. Držimo se *minimalističnega stila*. Uporabniku so v pomoč namigi, ki se pojavijo na vseh kontrolah, ko nad njih postavi miškin kazalec. Na voljo so tudi kratka navodila – *pomoč in dokumentacija*.

Pri zasnovi uporabniških vmesnikov so pogosto potrebni kompromisi med intuitivnim in natančnim, med izvedljivim in estetskim. Tak primer je v našem primeru izbira področja zanimanja. Intuitivno bi bilo, da uporabnik z miškinim kazalcem premika meje področja. Vendar je pri nastavljanju področja zanimanja potrebna natančnost na nivoju nekaj slikovnih elementov, zato smo se raje odločili za vrtljive kontrole, ki omogočajo premikanje meje z miškinim koleščkom za en slikovni element naenkrat.

3.3.4 Napake pri zaokroževanju

Včasih širina področja zanimanja ni deljiva z želenim intervalom, kar pomeni, da vrednosti ne moremo odčitati točno na mestu, kjer bi bilo to potrebno. Širina področja zanimanja je 4574 slikovnih točk. Graf prikazuje obdobje enega dneva. Če želimo odčitati intenziteto ob vsaki uri, razdelimo področje zanimanja na 24 delov, vsak od njih pa naj bi štel 190,583 slikovnih točk. Če v tem primeru zaokrožimo navzdol, bomo pri vsakem odčitku pridelali 0,583 slikovne točke napake in že v dveh intervalih bo to pomenilo več kot eno slikovno točko. Do konca grafa to pomeni že 14 slikovnih točk napake. Na sliki 3.7 je del grafa, kjer pride do take napake. Z modro barvo so označena mesta, kjer bomo odčitali meritev. Vidimo, da se razlika med modro črto in oranžnimi označbami na papirju povečuje. Pri visokih intenzitetah padavin ta zamik ni zanemarljiv.

Pri avtomatski različici programa ta napaka ni bila upoštevana. Širina intervala je bila vedno zaokrožena navzdol. Napako pri zaokroževanju pa lahko razmeroma preprosto zmanjšamo. Predhodno izračunamo, na koliko intervalov bomo pridelali eno slikovno točko napake in takrat ustrezno za-



Slika 3.7: Napaka pri odčitavanju se veča. Zadnji odčitek je 14 slikovnih točk (angl. pixel, px) za mestom, kjer bi se moral nahajati.

maknemo interval. To še ne pomeni nujno, da smo napako povsem odpravili. Pri večjem številu intervalov (na primer, ko odčitavamo vsakih 5 min in tako dobimo 720 intervalov) se lahko tudi majhne napake hitro seštejejo.

Poglavje 4

Rezultati

4.1 Kvalitativni

Za kvalitativno oceno avtomatskega postopka smo zaznano krivuljo primerjali z dejanskim potekom na grafu [1, 4]. Na vsakem grafu poiščemo točko, ki najbolj odstopa od pravilne vrednosti in določimo stopnjo napake:

- razred 0: napak ni ali pa so manjše od 0,2 mm
- razred 1: prišlo je do manjših napak, odstopanje med 0,2 mm in 0,5 mm
- razred 2: prišlo je do večjih napak, odstopanje med 0,5 mm in 10 mm
- razred 3: prišlo je do napak, ki povzročijo napačen izračun dnevnih padavin, odstopanje nad 10 mm

Vzroki za napake 3. razreda so zgrešene inverzije. Napake 1. in 2. razreda povzročijo madeži črnila, ki zmotijo potek krivulje. Napake razreda 0 so sprejemljive, saj do te mere lahko na rezultat vpliva že debelina črte. Kot je razvidno iz tabele 4.1, je bila z avtomatskim postopkom večina grafov (75,9 %) zaznanih brez večjih napak.

Namen interaktivnega postopka je omogočiti uporabniku, da ročno odpravi vse tri razrede napak. Padavinskim grafom lahko ročno dodajamo

Postopek	Razred 0	Razred 1	Razred 2	Razred 3
Avtomatski	44 (75,9 %)	6 (10,3 %)	3 (5,2 %)	5 (8,6 %)
Interaktivni	58 (100 %)	0	0	0

Tabela 4.1: Rezultati kvalitativnega testiranja avtomatskega postopka.

inverzije ter brišemo in rišemo krivuljo. Tako lahko z dovolj potrpežljivosti odpravimo vsako napako pri zaznavi krivulje, kot je razvidno iz tabele 4.1.

4.2 Kvantitativni

ARSO hrani podatke o padavinah. Za oceno natančnosti lahko primerjamo naše rezultate, prebrane iz grafa, z uradnimi podatki. Pri analizi rezultatov avtomatskega postopka so izvzeti rezultati grafov, kjer je prišlo do napak 3. razreda, saj niso reprezentativni – nujno potrebna bi bila ročna obdelava, ki v avtomatski različici ni možna. Takih je 5 grafov. Prav tako niso upoštevani grafi tistih dni, ko je bilo zabeleženega več kot 5 cm zapadlega snega. Zaradi topljenja snega ti podatki ne predstavljajo dejanskih padavin [1, 4]. Grafov s topljenjem snega je 8. Ostane 45 grafov, nad katerimi smo izvedli analizo. Rezultati so prikazani v tabeli 4.2.

Za oceno napake smo uporabili tri mere:

- srednja absolutna napaka (MAE) [5]:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |f(i) - \hat{f}(i)| \tag{4.1}$$

- srednja absolutna napaka izražena v odstotkih (MAEP) [5]:

$$\mathrm{MAEP} = \frac{N \times \mathrm{MAE}}{\sum_{i=1}^{N} f(i)} \tag{4.2}$$

- relativna srednja absolutna napaka (RMAE) [5]:

$$\mathrm{RMAE} = \frac{N \times \mathrm{MAE}}{\sum_{i=1}^{N} |f(i) - \bar{f}|} \tag{4.3}$$

Srednja absolutna napaka (MAE) nam pove, za koliko mm se v povprečju razlikujejo vrednosti. MAEP predstavlja to napako izraženo v odstotkih. Relativna srednja absolutna napaka (RMAE) pa nam pove relativno napako glede na razpon možnih vrednosti. Zavzame vrednost od 0 do 1, kjer 0 pomeni popolno ujemanje brez odstopanj.

Rezultati avtomatskega postopka so dobri – MAE je relativno majhen. Na dnevnem in urnem intervalu je postopek zelo uspešen, s krajšanjem intervala pa se napake seštevajo in pride do večjih razlik.

Enak postopek analize smo izvedli nad rezultati interaktivnega programa. Tudi tu smo iz analize izključili grafe, ki so bili izpuščeni iz analize avtomatskih rezultatov, čeprav jih sedaj lahko obdelamo. Pričakujemo, da bodo odstopanja od pravilnih vrednosti enaka ali celo manjša, saj smo potek grafov ročno popravili. Rezultati analize so prikazani v tabeli 4.3.

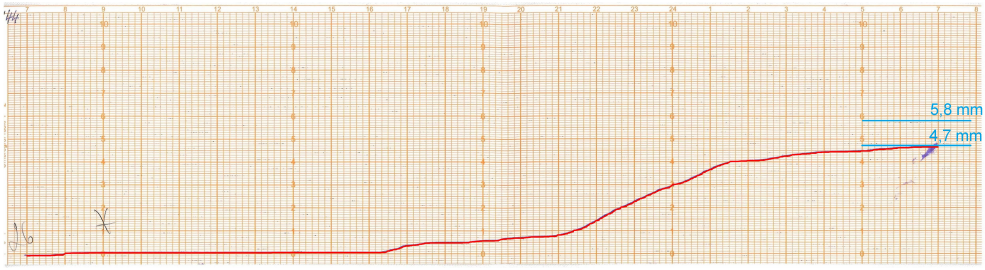
Časovna ločljivost	MAE [mm]	MAEP [%]	RMAE	N
dan	$0,3844$	$2,56$	$0,0321$	45
ura	$0,1350$	$8,55$	$0,0960$	394
pol ure	$0,1055$	$13,36$	$0,1379$	788
5 minut	$0,0563$	$42,77$	$0,3958$	4728

Tabela 4.2: Rezultati kvantitativnega testiranja za avtomatski postopek.

Časovna ločljivost	MAE [mm]	MAEP [%]	RMAE	N
dan	$0,4244$	$2,824$	$0,0354$	45
ura	$0,1170$	$7,41$	$0,0832$	394
pol ure	$0,0997$	$12,63$	$0,1304$	788
5 minut	$0,0573$	$43,56$	$0,4031$	4728

Tabela 4.3: Rezultati kvantitativnega testiranja za interaktivni postopek.

Ugotovimo, da je pri urnem in polurnem intervalu interaktivni postopek v



Slika 4.1: Graf padavin na dan 27. 2. 2006, obdelan z interaktivnim postopkom. Označeni sta vrednosti 4,7 mm in 5,8 mm.

skladu s pričakovanji uspešnejši od avtomatskega. Tako kot pri avtomatskem postopku je najuspešnejše dnevno odčitavanje (opazujemo relativne napake – RMAE in MAEP). S krajšanjem intervala se napaka veča. Kadar odčitavamo dnevno ali 5 minutno intenziteto padavin pa je bil avtomatski postopek na videz natančnejši. Na teh intervalih so grafi, ki smo jih ročno popravili, manj podobni uradnim podatkom, kot tisti, ki so bili zaznani avtomatsko. Vzrok za to so napake na odčitanih podatkih iz ARSO.

Napake na podatkih

Ogledali smo si nekaj največjih odstopanj med našimi rezultati in podanimi podatki na dnevnem intervalu. Odstopanja so bila velika – tudi preko 1 mm zaznanih padavin. Tako napako lahko hitro zaznamo s prostim očesom. Na sliki 4.1 je primer grafa, pri katerem je interaktivno dobljena dnevna intenziteta znašala 4,7 mm, v arhivih je hranjena vrednost 5,8 mm, avtomatski postopek pa je vrnil vrednost 4,8 mm. S slike je razvidno, da se krivulja konča na vrednosti 4,6 mm, prištejemo še 0,1 mm, saj se je merjenje začelo 0,1 mm pod vrednostjo 0. S prostim očesom lahko z gotovostjo trdimo, da sodeč po grafu na ta dan zagotovo ni zapadlo 5,8 mm, kot je zapisano v arhivu. Izkaže se, da so naši algoritmično dobljeni podatki pravilni, napaka je na strani arhivskih podatkov.

Ročno smo izmerili dnevno količino padavin za vse grafe. Če se je vre-

dnost v arhivu od naše razlikovala za vsaj 0,5 mm, smo jo popravili. Tako smo zamenjali 17 od 45 vnosov. Ponovno smo izvedli analizo tako avtomatskih kot interaktivnih rezultatov. Popravljeni rezultati so v tabeli 4.4. Vidimo, da je napaka v obeh primerih upadla, interaktivno dobljene vrednosti pa so sedaj bistveno bližje pravilnim vrednostim.

Časovna ločljivost	MAE [mm]	MAEP [%]	RMAE	N
dan (avtomatsko)	$0,1778$	$1,20$	$0,0152$	45
dan (interaktivno)	$0,16$	$1,08$	$0,0136$	45

Tabela 4.4: Rezultati kvantitativnega testiranja na dnevnem intervalu po odpravi napak na podatkih.

Popravljanje podatkov na dnevnem intervalu ni težavno, saj je meritev malo. Ročno popravljanje ostalih podatkov pa bi bilo časovno preveč potratno. Če bi popravili tudi ostale podatke, lahko predvidevamo, da bi zmanjšali napako še na vseh ostalih intervalih.

Pri 5 minutnem intervalu je velika omejitev tudi ločljivost vhodne slike. Odčitka sta v primeru slovenskih grafov oddaljena približno 15 slikovnih točk in težko dosežemo natančnost na takem nivoju.

Poglavje 5

Zaključek

Razvit je bil program za interaktivno branje padavinskih grafov, ki omogoča digitalizacijo padavinskih grafov s papirnih trakov. Tudi kadar je krivulja na grafu slabo razvidna lahko uporabnik z lastnoročnimi popravki pridobi natančne podatke o intenziteti padavin ter jih shrani za nadaljnjo analizo. Interaktivnost tako zagotavlja, da lahko pravilno zaznamo vsak graf, ne glede na njegov format in kvaliteto. To je glavna prednost programa, saj je postopek tako bolj prilagodljiv in uporaben na vseh padavinskih grafih.

Prejeli smo devet padavinskih grafov iz Čila, ki jih z novo različico programa lahko uspešno preberemo. Žal za te grafe nimamo podatkov zlatega standarda (angl. ground truth), s katerimi bi lahko naše rezultate primerjali. Poleg uporabe na širšem spektru grafov nam interaktivnost dokazano daje tudi boljše rezultate, čeprav je analiza zaradi pomanjkljivosti podatkov otežena.

Obstoječi avtomatski postopek je bil prenovljen z novimi knjižnicami in vgrajen v grafični uporabniški vmesnik, ki je do uporabnika prijaznejši in omogoča učinkovito odčitavanje padavinskih grafov. Tako je program pripravljen za uporabo v stroki. Vzpostavljeno je bilo tudi spletno mesto, kjer je poleg programa in dokumentacije na voljo izvorna koda. Program je zaščiten z odprtokodno licenco, ki omogoča izboljševanje in prilagajanje specifičnim potrebam raziskovalcev.

Možnosti izboljšav

Glavno nalogo – branje padavinskih grafov program uspešno opravlja. Pri preizkušanju pa smo naleteli na nekaj idej, ki bi smiselno dopolnile izkušnjo in olajšale delo:

- Hitro, zaporedno procesiranje večjega števila slik. Voden postopek, ki bi naenkrat obdelal celotno mapo slik ter uporabniku prikazal rezultate, z možnostjo hitrih popravkov.
- Možnost shranjevanja trenutnega stanja obdelave in obnovitev stanja iz datoteke. Trenutno ob zaprtju programa izgubimo vse popravke. Če želimo ponovno izvoziti podatke iste slike v drugačni obliki, moramo postopek ponoviti.
- Vključenje optičnega branja slik v sam program. Tako bi program zajemal celoten postopek, od zajetja podatkov do končnega rezultata.
- Optimizacija hitrosti. Trenutni program ne izkorišča moči procesorjev z več jedri in nitenja. Postopek bi s paralelizacijo lahko izvedli hitreje, oziroma vsaj zmanjšali čas čakanja uporabnika.
- Širitev na več platform. Naš primarni cilj je delo na računalnikih z operacijskim sistemom Microsoft Windows. Vse knjižnice, ki smo jih uporabili pa nudijo podporo za več platform, zato bi tudi naš izdelek lahko z nekaj dodatnega dela deloval še na vseh ostalih operacijskih sistemih.

Literatura

[1] Derganc, G., 2009. *Računalniško branje padavinskih grafov*. Diplomsko delo. Ljubljana: Univerza v Ljubljani, Fakulteta za računalništvo in informatiko.

[2] Galitz, W. O., 2002. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. 2. izdaja. New York: John Wiley & Sons, Inc.

[3] Hoffmann, G. 2013. *CIELab Color Space*. Dostopno na: http://docs-hoffmann.de/cielab03022003.pdf (15. 8. 2015).

[4] Jaklič, A., Šajn, L., Derganc, G., Peer, P., 2015. Automatic digitization of pluviograph strip charts, *Meteorological Applications*, sprejeto v objavo.

[5] Kononenko, I., 1997. *Strojno učenje*. Ljubljana: Fakulteta za računalništvo in informatiko.

[6] Nielsen, J. 1995. *10 Usability Heuristics for User Interface Design*. Dostopno na: http://www.nngroup.com/articles/ten-usability-heuristics/ (15. 8. 2015).

[7] Pavšič, J., 2006. *Geološki terminološki slovar*. Ljubljana: Založba ZRC, ZRC SAZU.

[8] *Boost Library Documentation: Date Time*. Dostopno na: http://www.boost.org/doc/libs/1_59_0/doc/html/date_time.html (15. 8. 2015).

[9] GIT repozitorij programa za interaktivno branje padavinskih grafov. Dostopno na:
https://github.com/nejc-susin/rain-read (13. 9. 2015).

[10] GNU General Public Licence. Dostopno na:
https://www.gnu.org/copyleft/gpl.html (15. 8. 2015).

[11] *LTI-Lib On-Line Manual*. Dostopno na:
http://ltilib.sourceforge.net/doc/html/index.shtml (15. 8. 2015).

[12] *OpenCV API Reference*. Dostopno na:
http://docs.opencv.org/modules/refman.html (15. 8. 2015).

[13] *Qt Documentation*. Dostopno na: http://doc.qt.io/ (15. 8. 2015).