UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Gašper Derganc

Računalniško branje padavinskih grafov

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Peter Peer

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01540/2009
Datum: 15.01.2009

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

Kandidat: **GAŠPER DERGANC**

Naslov: **RAČUNALNIŠKO BRANJE PADAVINSKIH GRAFOV**
COMPUTER READING OF RAINFALL GRAPHS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Avtomatske meteorološke merilne postaje zapisujejo količino padavin na papirnate trakove ter tako ustvarijo padavinske grafe. Meteorologi nato te grafe ročno pregledajo in iz njih izluščijo pomembne podatke za nadaljno obdelavo. Implementirajte rešitev na osnovi računalniškega vida, ki bo takšne padavinske grafe samodejno brala, iz njih pridobila ustrezne podatke. Ob snovanju preučite tudi sorodne rešitve. Naredite skrbno analizo učinkovitosti razvite rešitve, ki naj pokaže, pod kakšnimi pogoji dosegamo določeno natančnost.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Franc Solina

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Gašper Derganc,

z vpisno številko 63990174,

sem avtor diplomskega dela z naslovom:

Računalniško branje padavinskih grafov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Peter Peer
- 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 30.3.2009 Podpis avtorja/-ice:

Zahvala

Na tem mestu bi se rad zahvalil mentorju doc. dr. Petru Peeru, za pomoč in koristne nasvete, gospodu Zorku Vičarju z ARSO za podatke, padavinske grafe ter koristne razgovore, ki so mi dali širši vpogled v problemsko domeno naloge.

Kazalo

Povzetek

V tem delu je predstavljena metoda za avtomatično detekcijo in digitalizacijo krivulje padavin v padavinskih grafih s papirnatih trakov, ki se jih uporablja v avtomatskih merilnih postajah. Metoda sestoji iz več korakov. Na digitalni sliki grafa padavin se krivuljo padavin loči od ozadja. S sodelovanjem metod drsečega povprečja ter sledenja roba krivulje se krivulja detektira ter edinstveno določi – vsakemu stolpcu slike ustreza natanko ena točka. Ta detektirana krivulja pa je vhod v proces izdelave natančnega časovnega zaporedja padavin. Poleg postopkov analize slik se metoda opira tudi na mehanske značilnosti merilnega instrumenta. Natančno časovno zaporedje padavin je potrebno za nadaljnje analize padavinskih dogodkov kot so klasifikacija, analiza ekstremnih dogodkov, kalibracija modelov odtoka površinskih voda, napovedovanje meteoroloških pojavov ter v mnogih raziskovalnih projektih. Algoritem je bil preizkušen na 58 slikah pluviografskih trakov. Primerjava med rezultati pridobljenimi z opisanim algoritmom ter uradnimi podatki z Agencije Republike Slovenije za okolje je pokazala, da algoritem večinoma zelo natančno določi potek krivulje in s tem natančno časovno zaporedje padavin. Ker pa ni vedno 100% zanesljiv, ga je potrebno vključiti v sistem, ki bi omogočal ogled ter morebitno interaktivno spreminjanje delov potrebnih popravljanja.

Ključne besede:

računalniški vid, digitalizacija, padavinski graf, meteorologija, padavine

Abstract

An algorithm aimed at automatic detection and digitalisation of the rainfall signal recorded by float based rain gauges on paper strip charts is presented. The algorithm consists of several steps that gradually lead to the desired goal. The rainfall signal is extracted from the digital image of the strip chart. By cooperation of moving average method and curve edge following method the rainfall curve is detected and uniquely determined – in each image column there is one single point representing the rainfall curve plotline. From the curve plotline high resolution rainfall time series is obtained. Beside image analysis techniques in the design of the algorithm, the mechanical features of the recording instrument have been taken into consideration. The availability of high resolution rainfall time series is required in many applications, including rainfall classification, analysis of extreme rainfall events, calibration of raifall-runoff models, weather prediction models and many research projects. The algorithm has been tesed on 58 pluviograph strip chart images. The comparison between the data obtained by the proposed algorithm and the official data from the Environmental Agency of the Republic of Slovenia shows that the algorithm usually accurately detects the rainfall curve and consequently an accurate rainfall time series is obtained. Since it is not always 100% reliable it should be used as a component of a system that would enable the inspection of the detected curve and when required it should also enable interactive changing of the parts needing correction.

Keywords:

computer vision, digitalisation, rainfall graph, meteorology, rainfall

Poglavje 1

Uvod

Hidrologija je znanost, ki se ukvarja s pojavom, porazdelitvijo in gibanjem vode po Zemlji, vključno z vodo v atmosferi ter podzemsko vodo. Razen podzemske vode je zemeljska vodna zaloga v nenehnem kroženju s površine Zemlje v atmosfero in nazaj. Temu kroženju pravimo hidrološki cikel. Ta opisuje procese padavin, potovanja zapadle vode po rekah v jezera, oceane ter izhlapevanja nazaj v atmosfero. Ti so med seboj seveda odvisni – tako natančnejše poznavanje enega dela ponudi boljši vpogled v celotno dogajanje v ciklu.

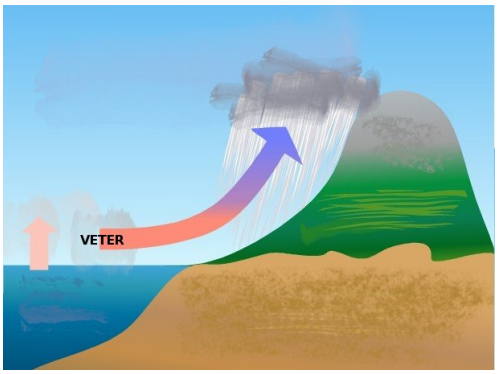
O padavinah govorimo v primeru padanja vode z atmosfere na Zemljino površino, bodisi v tekočem, bodisi trdnem agregatnem stanju. Za nastanek padavin so bistveni naslednji dejavniki:

- Mehanizem, ki shladi zrak in s tem povzroči kondenzacijo ter rast kapljic. Do ohlajanja zraka navadno pride ob dvigovanju le tega.

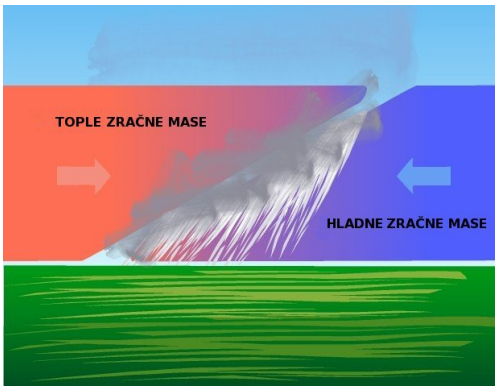
- Za nastanek kapljic so pomembna kondenzacijska jedra v obliki mikroskopskih delcev, ki so vedno prisotni v atmosferi.

Najpogostejši vzroki za nastanek padavin so prikazani na sliki 1.1.

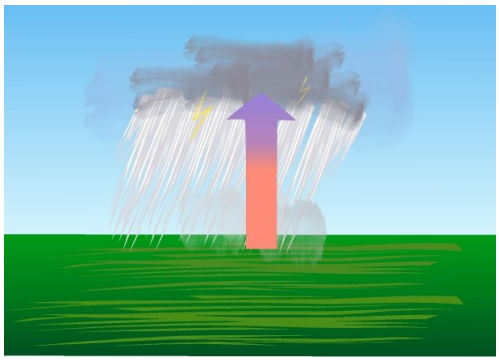
Vlaga v zraku prehaja v drobne kapljice vode ali ledu (oblaki), ki se med seboj lepijo ter povečujejo, dokler ne dosežejo take mase, da se začnejo pod vplivom gravitacije izločati v različnih oblikah padavin. V zmernih geografskih širinah v višjih delih oblaka vedno sneži, kakšne padavine bodo prispele do tal, pa je odvisno od temperaturnih in vlažnostnih razmer [1, 10].



(a) Orografske padavine.



(b) Ciklonske padavine.



(c) Konvekcijske padavine.

Slika 1.1: Različni vzroki za nastanek padavin.
Orografske padavine nastanejo, kadar vetrovi dvignejo vlažen zrak čez orografske ovire (gorske pregrade). Zrak se ohlaja, posledica so padavine.
Ciklonske padavine nastanejo ob prehodu tople in hladne fronte, ko se mora topli zrak, ki je lažji, dvigniti nad hladnega.
Konvekcijske padavine nastanejo, če se zemeljsko površje segreje, zrak se hitro dviga v ozračje in se adiabatno ohlaja.

1.1 Zajemanje meteoroloških podatkov

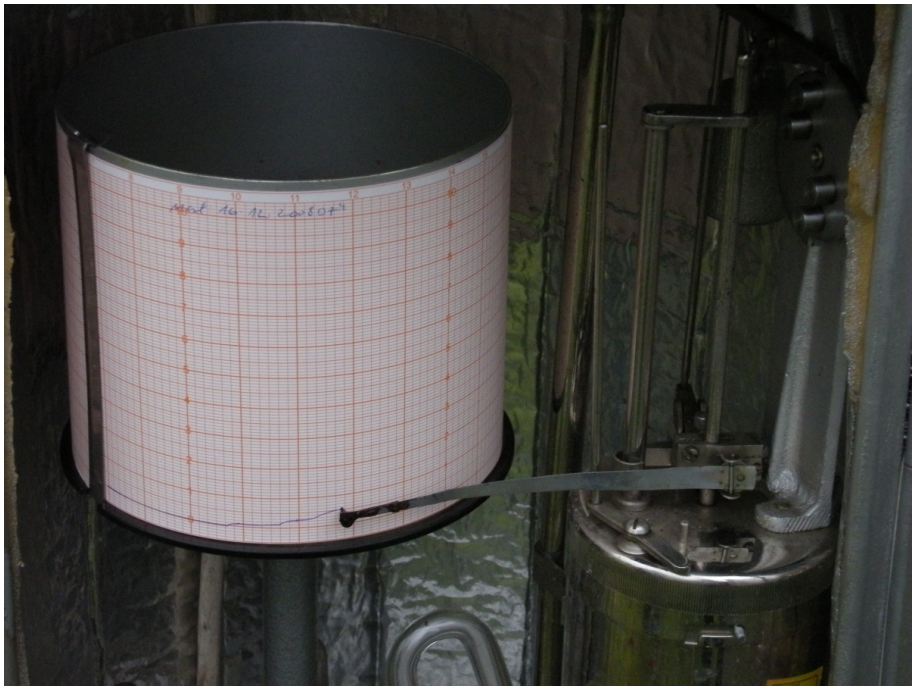
Količino padavin merimo kot višino vodnega stolpca, ki se akumulira na določeni horizontalni površini v določenem časovnem intervalu. Navadno je izražena v milimetrih, kjer 1 mm padavin ustreza 1 kg/m^2 oz. povedano drugače, če zlijemo 1 kg vode po površini enega kvadratnega metra, bo višina vodne plasti enaka enemu milimetru. Padavine merimo s pomočjo ročnih (pluviometri) ali avtomatskih (pluviografi) instrumentov. Pluviometri za razliko od pluviografov ne podajajo spreminjanja količine padavin v času. Omogočajo le ročno beleženje količine padavin v nekem časovnem intervalu (običajno 24 ur) [1].

Slika 1.2: Enostaven pluviometer.

- **Pluviometri**:
  Za merjenje padavin je uporabna vsaka cilindrična posoda z vertikalnimi stenami. Zaradi primerljivosti rezultatov se uporablja standardizirane posode. Pri nas je v uporabi Helmanov pluviometer s posodo volumna 200 cm^2. Tako so tudi napake merjenja na različnih postajah enake magnitude. Podajajo le podatek o celotni količini padavin v nekem

(a) Pluviograf s plovcem.

(b) Vhodna odprtina.



(c) Notranjost.

Slika 1.3: Pluviograf s plovcem.

časovnem intervalu. Običajno se dnevno (ob 7:00) ročno izmeri količina padavin, ki je padla v 24 urah (slika 1.2).

- **Pluviografi**:
  Beležijo spreminjanje količine padavin v času bodisi na papir ali v digitalni obliki. Tu bom opisal pluviografe, ki rezultate zapisujejo na papirnate trakove. Glede na način delovanja ločimo naslednje vrste pluviografov:

  1. Pluviograf-tehtnica:
     Padavine se zbirajo v posodi. Sprememba teže posode vpliva na premikanje pisala po vrtečem se kolutu papirja. Posoda se, ko se napolni, avtomatsko izprazni.
  2. Pluviograf z zlivajočima se posodicama (angl. tipping bucket):
     Pluviograf tega tipa vsebuje dve majhni posodici. Padavine, ki vstopajo v merilno napravo se zbirajo v prvi posodici. Ko se ta posodica napolni, se izprazni, pisalo se pomakne za določen korak (gor ali dol) po vrtečem se papirnatem valju, posodica pa se zamenja z drugo. Ko pisalo doseže zgornji ali spodnji rob traku se smer pomikanja pisala ob praznjenju posodic obrne. Pridobljena krivulja padavin ni gladka.
  3. Pluviograf s plovcem:
     Ko se voda dviga, plovec na površini premika pisalo po vrtečem se papirnatem valju. Ko se posoda napolni, se hitro izprazni preko avtomatskega odtoka. Prikazan je na sliki 1.3.
  4. Radarski pluviograf:
     Za merjenje količine padavin se lahko uporablja tudi mikrovalovni radar z valovno dolžino od 1 do 20 cm. Energija odbitih valov je sorazmerna velikosti kapljic ter s tem tudi intenziteti padavin.

  Rezultat meritev s pluviografom je funkcija količine padavin v odvisnosti od časa.

Intenziteta padavin je predstavljena kot količina padavin P na enoto časa t:

$$\text{intenziteta} = \frac{\mathrm{dP}}{\mathrm{dt}} \approx \frac{\Delta \mathrm{P}}{\Delta \mathrm{t}} \tag{1.1}$$

Funkcija, ki podaja intenziteto padavin v odvisnosti od časa, je navadno predstavljena s stolpčnim diagramom.

1.2 Opis problema

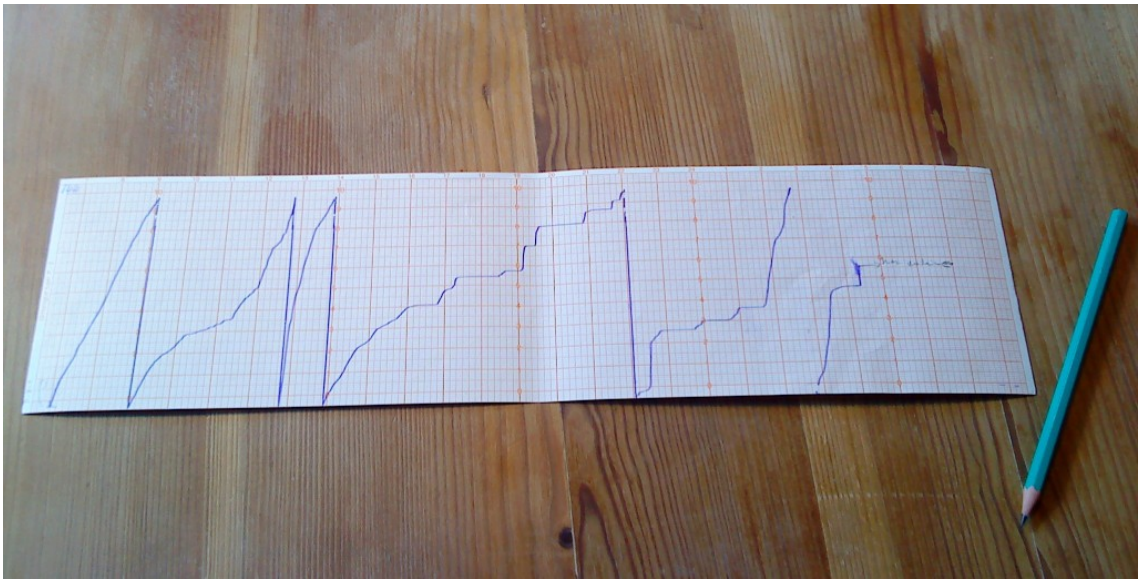
Dandanes se podatke o padavinah ponavadi zbira z avtomatskimi digitalnimi pluviografi. Nekateri beležijo čas v katerem je zapadla določena količina padavin, medtem ko drugi shranjujejo podatek o kumulativni količini padavin v diskretnih časovnih intervalih. V Sloveniji se večinoma uporabljajo digitalni pluviografi z zlivajočima se posodicama (angl. tipping bucket). Prednost teh instrumentov so enostavno dostopni podatki v digitalni obliki, ki so takoj na voljo za nadaljne računalniške obdelave.

Pomanjkljivosti so:

- Natančnost je odvisna od velikosti posodice, ki se giblje od 0,2 mm do 1 mm.
- Pri intenzivnih padavinah lahko zaradi premikanja posode pride do izgube padavin.
- Pri manjših padavinah na meritve vpliva izhlapevanje iz posodic.
- Pri padavinah z nizko intenziteto ni mogoče določiti časa začetka ali prenehanja padavin.
- Ob izpadu električne energije (npr. v primeru udara strele) ne deluje.

Pred uvedbo digitalnih avtomatskih pluviografov so se podatki avtomatskih pluviografov zapisovali le na papirnate trakove (slika 1.4). V Sloveniji trenutno deluje 38 merilnih mest, s pluviografi s plovcem, ki tako shranjujejo rezultate svojih meritev. Ti trakovi na osi x prikazujejo čas (24 ur – od 7:00 do 7:00) ter na y osi podatek o polnosti posode (od 0 do 10 mm). Ko se posoda napolni, se izprazni, kar je na traku vidno kot hiter padec krivulje do vrednosti 0 po osi y. Podatki s teh postaj so koristni za preverjanje podatkov z avtomatskih digitalnih merilnih postaj ter natančnejšo analizo padavin iz preteklih obdobij, ko avtomatske digitalne merilne naprave še niso bile v uporabi. Koristni pa so tudi zaradi svoje robustnosti, saj so digitalni pluviografi nagnjeni k napakam prav v primerih ekstremnih vremenskih pojavov, ki so za meteorologe ter hidrologe najbolj zanimivi.

Tudi te podatke je potrebno pretvoriti v računalniku prijazno obliko. Ta postopek se izvaja s pomočjo digitalizatorske table ter zahteva veliko zbranosti in natančnosti ter je časovno zelo zahteven. Papirnati trak se položi na tablo, označi se robove področja zanimanja ter določi potek krivulje z označevanjem



Slika 1.4: Trak pluviografa.

točk krivulje. Iz pridobljenega zaporedja točk, katerih vmesne vrednosti so določene z linearno interpolacijo, se s pomočjo programa izračuna intenziteta padavin. Za takšno obdelavo mesečnih podatkov s posamezne merilne postaje, je potrebnih od 10 minut do 1 človek-ura, odvisno od na trakovih zabeležene količine padavin. Čas obdelave je torej močno odvisen tudi od geografske lege merilne postaje, s katere izvirajo trakovi – razlika povprečnih letnih padavin med postajami (npr. 744 mm (Veliki Dolenci) ter 2261 mm (Stara Fužina)) je namreč lahko zelo velika.

1.3 Cilji

Zaradi svojih pozitivnih lastnosti pluviografi s papirnatimi trakovi verjetno še dolgo ne bodo šli iz uporabe. Pridobitev natančnejših padavinskih časovnih zaporedij visoke resolucije s starejših pluviografskih trakov pa bi omogočilo boljši vpogled v preteklo padavinsko dogajanje. Trenutni postopek digitalizacije je zelo zamuden ter monoton. S pomočjo v tem delu predlaganega postopka bi ga bilo mogoče občutno pospešiti ter morda tudi povečati natančnost tako pridobljenih podatkov.

Namen tega dela je predstaviti postopek, ki bo:

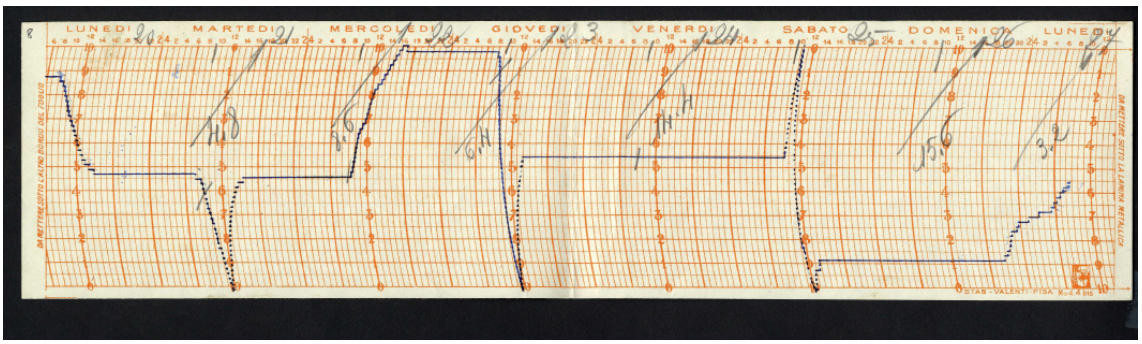
- pospešil obstoječi postopek,
- čim natančneje določil potek krivulje,
- robusten – vrnil razmeroma dobre rezultate tudi v primeru slabo zaznavnega signala padavin,
- omogočil ogled dobljenih rezultatov,
- omogočil shranjevanje rezultatov v željeni obliki.

Pri implementaciji postopka sta bili uporabljeni dve neodvisni množici slik trakov pluviografov. Implementacija algoritma je bila izvršena s pomočjo učne množice. Učna množica vsebuje 8 trakov z merilne postaje Kal nad Kanalom. Za namene testiranja je bilo uporabljenih 58 slik trakov z merilne postaje Podkraj, ki so naključno izbrane izmed slik trakov iz leta 2006. Za analizo rezultatov so bili na voljo podatki o dnevnih, urnih, polurnih ter 5 minutnih intenzitetah padavin z merilne postaje Podkraj, kot so jih zabeležili na Agenciji Republike Slovenije za okolje.

Poglavje 2

Sorodna rešitev

V tem poglavju je predstavljen postopek za digitalizacijo padavinskih grafov, ki so ga na univerzi Cagliari v Italiji razvili Roberto Deidda, Giuseppe Mascaro, Enrico Piga ter Giorgio Querzoli. Namenjen je digitalizaciji padavinskih podatkov s pluviografov z zlivajočima se posodicama [2].



Slika 2.1: Primer traku s pluviografa z zlivajočima se posodicama.

Bistveni koraki postopka so:

- **Predprocesiranje**: nad vhodno sliko se izvedejo transformacije ki omogočajo predstavitev krivulje v kartezičnem koordinatnem sistemu.
- **Segmentacija**: loči slikovne elemente krivulje od ozadja ter rezultat shrani v binarno matriko.
- **Avtomatična detekcija signala**: v tem koraku se z metodo robustne detekcije krivulje (angl. Robust line detection method) enolično določi krivuljo ter, če je potrebno, izvede popravke nad pridobljenim časovnim zaporedjem.

- **Interaktivno postprocesiranje**: preko grafičnega vmesnika je mogoče preveriti ter ob morebitnih napakah postopka tudi popraviti pridobljene rezultate.

Predprocesiranje se izvaja nad digitalno sliko traku pluviografa. Izvede se rotacija, ki sliko poravna s skalo označeno na traku. Potrebno je opraviti tudi geometrično ukrivljanje slike (angl. warping), saj se pisalo obravnavanih merilnih naprav v smeri vertikalne osi ne premika linearno, temveč potuje po krožnici. Tako je krivulja lahko predstavljena v kartezičnem koordinatnem sistemu, kjer točke slike z enako vrednostjo abscise ustrezajo istemu časovnemu trenutku.

Korak segmentacije vsebuje upragovanje R komponente barvnega prostora RGB vhodne slike. Nato pa se izvede še nehierarhično rojenje (angl. non-hierarchical cluster analysis) v HSV prostoru barv, katerega rezultat sta dva razreda. Prvi vsebuje slikovne elemente krivulje, drugi pa slikovne elemente, ki so posledica pisanja s svinčnikom in niso zanimivi za nadaljnjo obdelavo. Mnogo papirnatih trakov je namreč polnih raznih zapiskov ter opomb, ki se nanašajo na zabeležene padavine.

V koraku avtomatične detekcije signala se iz segmentirane slike enolično določi potek krivulje. Za delovanje postopek potrebuje podatke o debelini črte ter lego in mere področja zanimanja. Korak je sestavljen iz petih procedur, ki upoštevajo mehanske lastnosti merilnega instrumenta. Te so:

- robustna detekcija krivulje,
- zavračanje madežev,
- omejitev na monotona zaporedja,
- popravki in prilagoditve detektirane krivulje,
- iskanje točk inverzije.

Vhod v postopek je digitalna slika papirnatega traku v znani resoluciji 300 DPI (slika 2.1). Bistvene razlike med temi trakovi ter trakovi s pluviografa s plovcem so:

- prekinjena krivulja – pisalo se vertikalno pomika po diskretnih intervalih, ki so določeni s prostornino posamezne posodice,
- os y v primeru trakov s pluviografov z zlivajočima posodicama ne predstavlja polnosti posode (ni praznjenja). Ko pisalo doseže rob, le zamenja smer pomikov,

- opazna ukrivljenost skale in krivulje, ki je posledica krožnega gibanja pisala.

Zaradi podobnosti problemov, sta osnovni zgradbi v tem delu opisanega algoritma ter algoritma, ki je opisan v tem poglavju, skoraj enaki. Razlikam botruje predvsem drugačen vhodni format slike ter različne odločitve pri reševanju problemov.

Predlagani algoritem v koraku segmentacije pretvori sliko v prostor barv CIELAB ter nad posameznimi komponentami slike določa točke slike, ki pripadajo krivulji s postopkom rasti regij. V tem poglavju predstavljeni algoritem izvaja upragovanje v prostoru barv RGB nato pa še nehierarhično rojenje v HSV prostoru barv.

Uporabljena metoda robustne detekcije je podobna metodi dvo-stopenjskega drsečega povprečenja, ki se v predlaganem algoritmu, poleg metode sledenja roba krivulje, uporablja v koraku detekcije. Sledenje roba krivulje v tem primeru (zaradi prekinitev krivulje) ne bi bilo primerno.

Tudi v koraku naknadne obdelave se oba algoritma spopadata s podobnimi problemi, le da so rešitve prilagojene posameznemu formatu zapisovanja na trak.

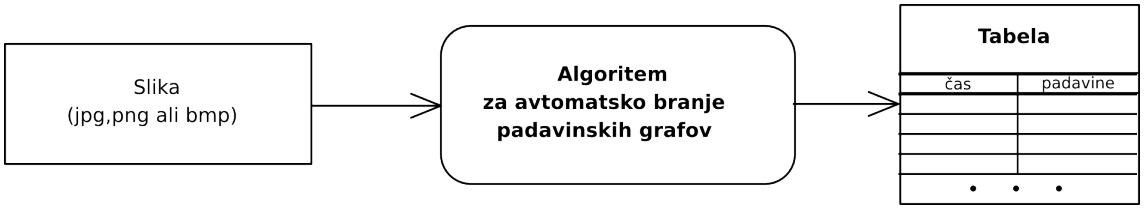
Poglavje 3

Algoritem za avtomatsko branje padavinskih grafov

V tem poglavju je opisan postopek, ki omogoča avtomatsko shranjevanje padavinskih podatkov s pluviografskih trakov (slika 3.1). Predstavljeni so vsi sestavni deli programa in navedeni razlogi, ki so botrovali izbiri algoritmov ter njihovih parametrov. Prikazani so različni tipi problemov, ki postopek otežujejo.

Osnovna ideja algoritma:

1. Loči krivuljo od ozadja – določi katere točke slike pripadajo krivulji.
2. Določi natančen potek krivulje – zaporedje točk, ki pokrivajo celotno dolžino traku z natanko eno točko na stolpec.
3. Iz koordinat točk izračunaj količino vode v zbiralni posodi ter iz razlik teh količin sosednjih točk določi intenziteto padavin.



Slika 3.1: Algoritem za avtomatsko branje padavinskih grafov.

Vhod v algoritem je slika v formatu JPG, PNG ali BMP. Barvna paleta slike je RGB z barvno globino 24 bitov na slikovni element. V učni in testni

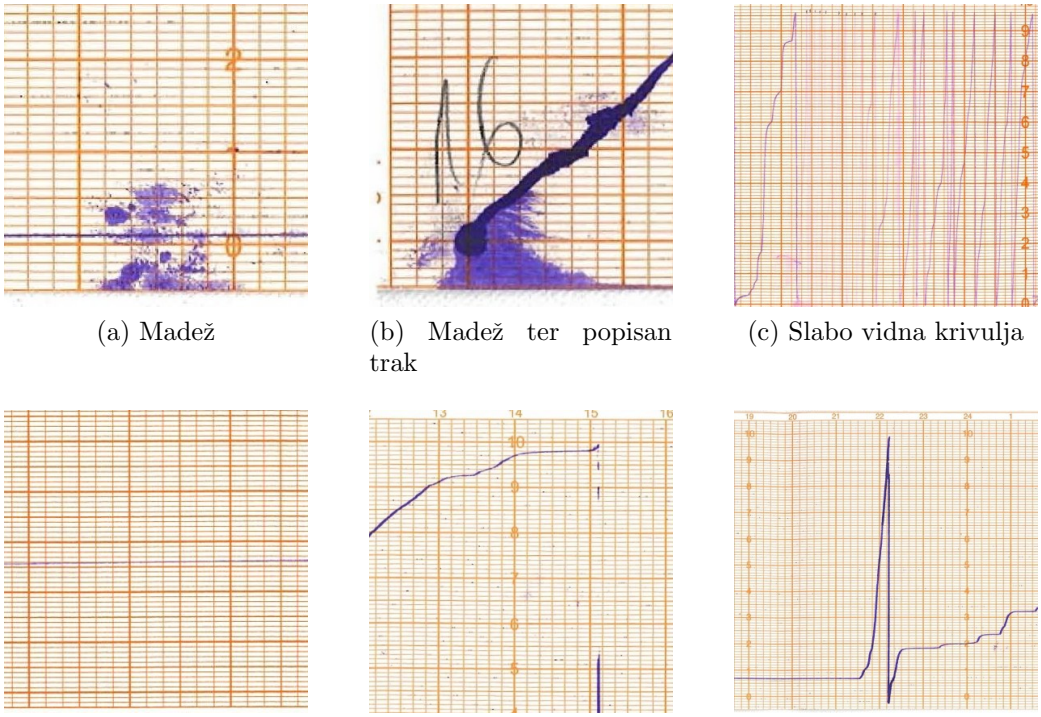
množici slik je uporabljena resolucija 300 DPI, ki je neke vrste kompromis med natančnostjo in časom izvajanja algoritma. Resolucija ni nujno fiksna, saj se ji algoritem lahko prilagodi z nastavitvijo parametrov. Pomembo je le, da ta ni prenizka, saj se z zmanjševanjem resolucije manjša tudi količina informacije slike. Za pravilno delovanje mora vhodna slika izpolnjevati določene pogoje:

- Slika je poravnana. Vse slikovne točke stolpca i ustrezajo istemu časovnemu intervalu. Poleg rotirane slike je vzrok za neporavnanost lahko tudi nenatančno vstavljen ali po vstavitvi premaknjen trak.
- Na grafu je le ena krivulja (v primeru dni brez dežja se včasih uporabi isti trak, kar rezultira v več krivuljah).
- Uporabljeno je modro črnilo pisala.

Izhod algoritma je tekstovna datoteka, v kateri so podatki ločeni s tabulatorji. Za vsak časovni interval je v svoji vrstici zabeležen čas začetka intervala in količina v tem intervalu izmerjenih padavin. Ta vrsta izhoda je izbrana, ker je takšno datoteko enostavno uvoziti v poljubno aplikacijo ali podatkovno bazo.

3.1 Vzrok šuma na vhodnih podatkih

Ker govorimo o padavinskih grafih, se povsem razumljivo z njimi pogosto upravlja v času dežja. Ravno dež pa je glavni krivec za večino nepravilnosti, ki jih lahko opazimo na trakovih. Nastanejo madeži, krivulja se razmaže ali v skrajnem primeru zabriše. Če želimo dobiti dobre rezultate, je treba upoštevati vse te faktorje in čim bolj omiliti njihove negativne vplive. Na slikah 3.2 je prikazanih nekaj tipičnih pojavov šuma ((a) in (b)), slabo vidne krivulje ((c), (d)) ter dva primera inverzij ((e) in (f)). Inverzije ne predstavljajo motilnega elementa in so pogosto prisotne. Na sliki 3.2 so prikazane, ker predstavljajo posebnost in jih je potrebno obravnavati ločeno.



(a) Madež (b) Madež ter popisan trak (c) Slabo vidna krivulja

(d) Slabo vidna krivulja (e) Pretrgana krivulja (f) Inverzija

Slika 3.2: Prikaz najpogostejših težav.

3.2 Koraki algoritma

Slika 3.3: Prikaz korakov algoritma.

Algoritem je sestavljen iz več korakov. Vsak korak predstavlja zaključeno celoto s tem, da reši določen podproblem, katerega rešitev je potrebna v naslednjem koraku. Bistveni koraki so prikazani na sliki 3.3. Ti so:

- **Segmentacija**: S pomočjo transformacije barv slike v prostor barv CIELAB se loči točke krivulje od točk ozadja.

- **Kalibracija**: Določi se področje zanimanja slike, ki nam kasneje pomaga prevesti koordinate točk v intenziteto padavin.
- **Detekcija krivulje**: Poiščemo zaporedje točk, ki enolično določa krivuljo (največ eno točko na stolpec slikovnih elementov slike).
- **Naknadna obdelava**: Poskrbimo, da zaporedje točk ustreza določenim omejitvam. Poskusimo odpraviti morebitne napake koraka detekcije. Interpoliramo neznane vrednosti.
- **Izpis rezultatov**: Izpišemo izmerjene intenzitete padavin v datoteko.

3.3 Segmentacija

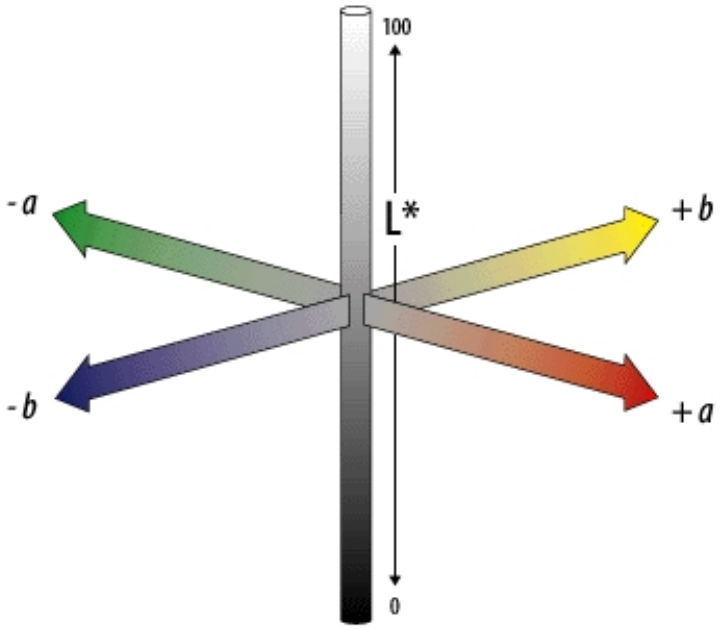
Namen segmentacije je čim bolje ločiti zapisano krivuljo od ozadja. Kot rezultat dobimo binarno sliko, kjer imajo slikovni elementi, ki so prepoznani kot del krivulje vrednost 1, ostali pa vrednost 0. Zaradi značilnosti ozadja in barve črnila je primerna uporaba nelinearnega prostora barv CIELAB.

3.3.1 Prostor barv CIELAB

Prostor barv CIELAB pokriva celoten vidni spekter barv in je zasnovan tako, da so prehodi med barvami, za človeško oko, kar se da enakomerni. Za razliko od RGB in CMYK je CIELAB namenjen čim boljšemu oponašanju človeškega vida. Spada v razred uniformnih prostorov barv, kjer je razdalja med dvema točkama v koordinatnem sistemu barv dober pokazatelj razlike med njima – v primeru, da je razdalja med barvama (v CIELAB koordinatnem prostoru) pod nekim pragom, sta barvi za opazovalca enaki.

Koordinatni sistem barv CIELAB temelji na osnovi nasprotujočih si barv: določena barva ne more biti rdeče-zelena kot tudi ne modro-rumena. Tako pozitivne vrednosti na osi a^* določajo količino rdeče, negativne pa količino zelene barve. Na b^* osi pozitivne vrednosti pripadajo rumeni, negativne pa modri barvi. Os L^* določa svetlost, ki tako kot ostali dve osi perceptualno enakomerno porazdeljuje svoje vrednosti (slika 3.4).

Da se izognemo dvoumnosti, je potrebno povedati, da se okrajšava CIELAB nanaša na barvni prostor CIE 1976 (L^*, a^*, b^*) in ne na starejši Hunter 1948 (L, a, b). Barvna prostora sta sicer ustvarjena z istim namenom (oponašanje človeškega vida), razlikujeta pa se v implementaciji. Z enakim namenom so imena koordinatnih osi označena z dodatno zvezdico ($*$). Sledijo enačbe nelinearne transformacije RGB koordinat v koordinate CIELAB [3, 12].



Slika 3.4: Grafična predstavitev osi CIELAB barvnega prostora.

Barvni prostor XYZ je podan kot :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0\ 490 & 0\ 310 & 0\ 200 \\ 0\ 17697 & 0\ 81240 & 0\ 01063 \\ 0\ 000 & 0\ 010 & 0\ 990 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.1}$$

Pretvorba v CIELAB oziroma natančneje CIE 1976 (L^*, a^*, b^*) pa je sledeča:

$$\begin{bmatrix} L^* \\ a^* \\ b^* \end{bmatrix} = \begin{bmatrix} 116(f(\frac{Y}{Y_n}) - 16) \\ 500(f(\frac{X}{X_n}) - f(\frac{Y}{Y_n})) \\ 200(f(\frac{Y}{Y_n}) - f(\frac{Z}{Z_n})) \end{bmatrix}, \tag{3.2}$$

kjer je

$$f(t) = \begin{cases} t^{\frac{1}{3}} & ; \text{če } t > (\frac{6}{29})^3 \\ \frac{1}{3}(\frac{29}{6})^2 t + \frac{4}{29} & ; \text{sicer} \end{cases}. \tag{3.3}$$

Enačba (3.3) zagotovlja, da ima krivulja v točki $t = 0$ končno velik naklon. Vrednosti X_n, Y_n in Z_n prostora barv CIE XYZ predstavljajo trojček, ki definira referenčno belo točko (spodnji n pove, da gre za normalizirane vrednosti). Ker se vrednosti prostora CIELAB raztezajo le od 0 do 100 (s plavajočo ve-

jico), jih zaradi lažjega računanja raztegnemo na interval od 0 do 255 (diskretne vrednosti), kar se lahko shrani v eno pomnilno besedo (angl. byte).

Očitno je bila barva ozadja ter barva črnila izbrana z namenom izdelave čim večje razlike barv za opazovalca, saj ti grafi prvotno niso bili mišljeni za računalniško obdelavo. Za takšno obdelavo pa je potrebna prevedba v prostor barv CIELAB, ki poenostavi postopek ločevanja krivulje od ozadja – segmentacijo. Tej transformaciji se lahko izognemo z uporabo naprednejšega optičnega čitalnika, ki za razliko od običajnih, ki informacijo o barvi shranjujejo v obliki RGB koordinat, shranjujejo CIELAB koordinate.

3.3.2 Upragovanje po postopku rasti regij

(a) Originalna sivinska slika. (b) Slika po upragovanju z rastjo regij.

Slika 3.5: Primer upragovanja z rastjo regij.

Izbira slikovnih elementov, ki pripadajo krivulji, je izvedena s postopkom upragovanja. Upragovanje je implementirano kot rast regij (angl. region growing). Ideja za algoritem je razvidna v [7]. Kot seme so izbrani vsi slikovni elementi, ki presegajo nek visok prag, katerega naj bi ga dosegali le slikovni elementi krivulje. Kot kriterij za nadaljno rast se uporablja spodnji prag in standardno odstopanje soseščine trenutnega slikovnega elementa. Na ta način se za razliko od navadnega upragovanja, ki izbere vse slikovne elemente, ki so po vrednosti med dvema pragoma, upošteva tudi sosednost. Zmanjša se možnost napačne detekcije, zaradi uporabe standardnega odstopanja pa upošteva tudi

lokalne značilnosti krivulje, katere intenziteta lahko variira. Tako se poveča tudi natančnost. Primer uporabe upragovanja z rastjo regij je prikazan na sliki 3.5.

Postopek je razviden iz algoritma 3.1. Ta zaradi preglednosti kode obravnava le primer, ko je zgornji prag večji od spodnjega.

```
1  rgThreshold(input, lowerThr, upperThr){
2    output.fill(0)
3    visited.fill(0)
4    // čez vse slikovne elemente slike
5    for point in input do
6      if not visited.at(point) and input.at(point) > upperThr then
7        // slikovni element je primeren kot seme za rast
8        // regije (neobiskan ter presega prag)
9        growRegion(point, lowerThr)
10   return output
11 }
12
13 growRegion(input, seed, lowerThr){
14   queue.put(seed)
15   sum = input.at(seed)
16   number = 1
17   avg = 0
18   stdev = 0
19   while not queue.empty()  do
20     point = queue.get()
21     output.at(point) = 1
22     avg = sum / number
23     stdev = calculateStdDev(neighborhood(point, 3x3))
24     // pregledamo vse sosede
25     for tmp in neighborhood(point, 1) do
26       if not visited.at(tmp) then
27         // slikovnega elementa še nismo obiskali
28         if input.at(tmp) > max(lowerThr, avg-stdev) then
29           // vrednost slik. elem. ne izstopa preveč od povprečja,
30           //  ga dodamo v regijo ter nadaljujemo
31           // z njim kot semenom
32           queue.put(tmp)
33           number = number+1
34           sum = sum + input.at(tmp)
35         // poskrbimo, da ne bo večkrat obiskan
36         visited.at(tmp) = 1
37 }
```

Algoritem 3.1: Upragovanje kot rast regij.

Regija se prične pri slikovnem elementu, ki presega zgornji določeni prag. Širi se na sosednje slikovne elemente dokler bodisi vsi slikovni elementi okolice regije ne presegajo določenega spodnjega praga, bodisi "preveč" odstopajo od povprečne vrednosti regije.

Enačbi uporabljeni pri upragovanju:

- povprečje vrednosti regije:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{3.4}$$

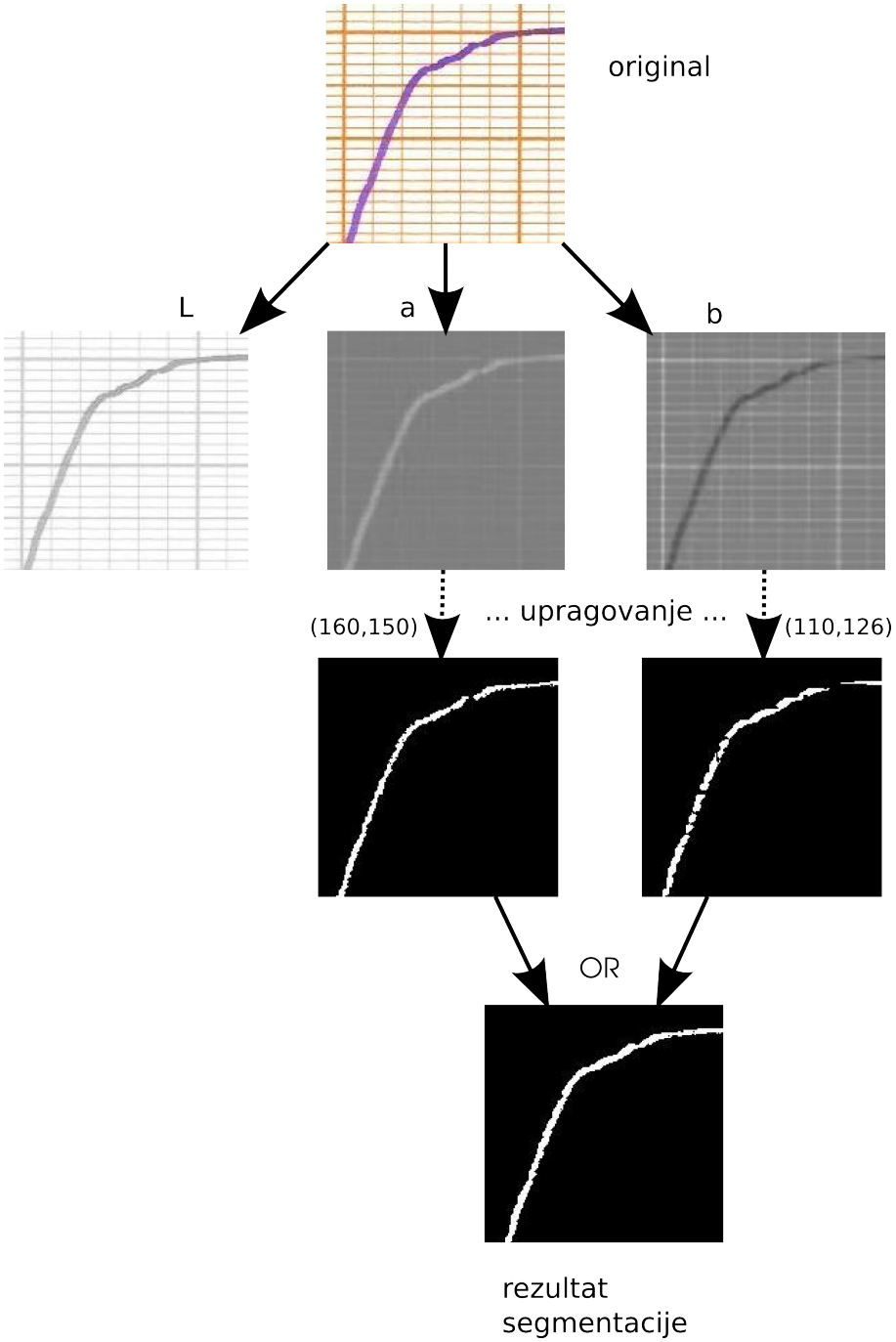
  kjer je x_i sivinska vrednost i-tega slikovnega elementa regije, ki vsebuje n slikovnih elementov

- standardno odstopanje:

$$\sigma = \sqrt{\frac{\sum_{j=1}^{n}(x_j - \bar{x})^2}{n-1}}, \tag{3.5}$$

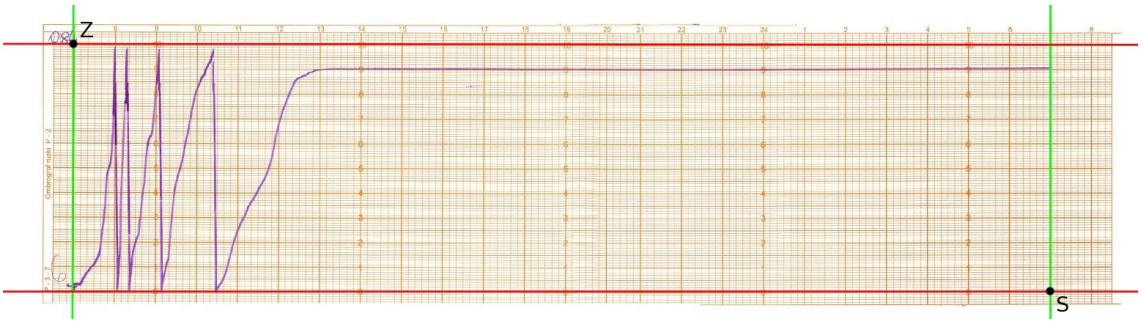
  kjer je x_j sivinska vrednost j-tega slikovnega elementa in n število vseh slikovnih elementov $N \times N$ okolice trenutnega semena.

Omenjeni postopek se uporabi na sivinskih slikah a^* in b^* (komponentah barvnega prostora CIELAB), kot je prikazano na sliki 3.6. Pred tem sta zglajeni z mediana filtrom [5] (z velikostjo okna 2×2), ki odpravi morebitne pomanjkljivosti uporabljenega optičnega čitalnika. Ko govorimo o sivinskih slikah imamo seveda v mislih slike, kjer se vrednost slikovnega elementa lahko shrani v nepredznačeno besedo (angl. byte) - torej obsega vrednosti od 0 do 255. Zgornja meja po vrednosti ni nujno večja od spodnje meje. Na sivinski sliki komponente a^* se rast regije prične pri slikovnem elementu, ki ima vrednost večjo od 160, spodnjo mejo pa določa vrednost 150. Pri komponenti b^* se rast regije prične pri slikovnem elementu z vrednostjo manjšo od 110. Spodnja meja pa je določena z vrednostjo 126. Vrednosti so določene kot posledica empiričnih izkušenj pridobljenih s pomočjo učne množico slik.



Slika 3.6: Koraki segmentacije.

3.4 Kalibracija



Slika 3.7: Področje zanimanja.

Da lahko kasneje detektirani potek krivulje opisane s seznamom točk slike prevedemo v podatek o intenziteti padavin, mora biti slika poravnana z na papirju označeno skalo. Poleg tega moramo poznati tudi področje zanimanja slike. To predstavlja ekstremne vrednosti krvulje, ki jih je mogoče enostavno razbrati s skale papirja. Le s poznavanjem področja zanimanja je mogoča enostavna transformacija koordinatnega sistema X-Y v koordinatni sistem čas-padavine. Področje zanimanja je torej pomembno za pravilen izračun padavin in ne omejuje iskanja krivulje, saj le ta lahko sega tudi izven tega področja – je zamaknjena. Področje zanimanja je definirano s pravokotnikom, ki ga omejuje. Tega pa lahko predstavimo z dvema točkama. $\mathbf{Z} = (x_z, y_z)$ je leva zgornja točka pravokotnika, $\mathbf{S} = (x_s, y_s)$ pa je spodnja desna točka. Tako lahko definiramo:

$$\text{višina področja zanimanja} = y_s - y_z \tag{3.6}$$

$$\text{širina področja zanimanja} = x_s - x_z \tag{3.7}$$

Področje zanimanja definirano s točkama **Z** in **S** je prikazano na sliki 3.7.

Pri iskanju točk **Z** in **S** si ponovno pomagamo s pretvorbo v prostor barv CIELAB, le da tu ne želimo izpostaviti slikovnih elementov krivulje temveč na traku označeno skalo. Natančneje – iščemo horizontalni črti, ki na skali predstavljata 0 mm in 10 mm (prazna, polna posoda) in vertikalni črti, ki na skali predstavljata začetek ter konec časovnega intervala merjenja (7:00 prvega in 7:00 naslednjega dne). Lokacija vseh štirih črt skupaj določi področje

zanimanja (komponente točk $\mathbf{Z}$ in $\mathbf{S}$). Lego teh točk bi lahko določili fiksno, v odstotkih mer slike, kar pa ne bi bilo dovolj fleksibilno. Pride lahko do napak pri branju z optičnim čitalcem, v primeru uporabe A3 optičnega čitalca je majhen del konca traku odrezan, možnih je več različnih formatov trakov, ki se relativno malo razlikujejo, ipd.

Področje zanimanja določimo z uporabo dveh pravokotnikov, ki približno določita lego meja področja zanimanja in tako omejita prostor iskanja. Določena sta relativno (v odstotkih mer slike) ter prav tako, kot področje zanimanja, definirana z dvema točkama:

- Zunanji pravokotnik: $\mathbf{R}_z = \{\mathbf{Z}_z = (x_{z_1}, y_{z_1}), \mathbf{S}_z = (x_{z_2}, y_{z_2})\}$:

$$\begin{aligned} \mathbf{Z}_z &= (\lfloor M * 0{,}015 \rfloor, \lfloor N * 0{,}050 \rfloor) \\ \mathbf{S}_z &= (\lfloor M * 0{,}960 \rfloor, \lfloor N * 0{,}990 \rfloor) \end{aligned} \tag{3.8}$$

- Notranji pravokotnik: $\mathbf{R}_n = \{\mathbf{Z}_n = (x_{n_1}, y_{n_1}), \mathbf{S}_n = (x_{n_2}, y_{n_2})\}$:

$$\begin{aligned} \mathbf{Z}_n &= (\lfloor M * 0{,}053 \rfloor, \lfloor N * 0{,}120 \rfloor) \\ \mathbf{S}_n &= (\lfloor M * 0{,}920 \rfloor, \lfloor N * 0{,}920 \rfloor) \end{aligned} \tag{3.9}$$

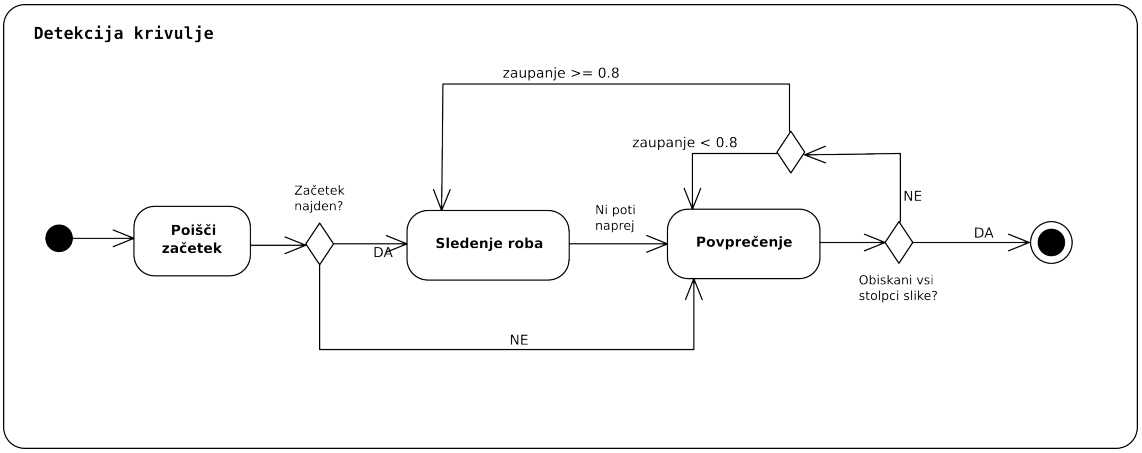
M in N sta meri slike velikosti $M \times N$. Zunanji pravokotnik $\mathbf{R}_z$, kot pove že ime, vsebuje celotno področje zanimanja, notranji $\mathbf{R}_n$ pa je v celoti vsebovan v področju zanimanja. Zanimajo nas le območja, ki jih omejujeta s svojimi stranicami – v teh se namreč nahajajo prej omenjene črte, ki omejujejo področje zanimanja. Uporabi se komponenta b^* prostora barv CIELAB, nad omenjenimi območji se izvede običajno upragovanje, ki loči točke skale od ozadja (zavržemo vse slikovne elemente, katerih vrednost je manjša od 145). Z glasovanjem poiščemo horizontalni in vertikalni črti. V primeru iskanja horizontalne premice določa lego te premice vrstica slikovnih elementov komponente b^* z največjo vsoto vrednosti teh elementov. Za vertikalni črti, namesto vrstic iščemo ustrezen stolpec. Lokacija vseh štirih črt nam skupaj določi področje zanimanja (komponente točk $\mathbf{Z}$ in $\mathbf{S}$ področja zanimanja). Pristop je neke vrste poenostavljena Houghova transformacija [11] za iskanje premic, kjer iščemo le horizontalne/vertikalne premice.

Zunanji ter notranji pravokotnik nam pomagata v koraku detekcije tudi pri iskanju začetka krivulje. Področje iskanja začetka krivulje: $\mathbf{O} = \{\mathbf{Z}_o, \mathbf{S}_o\}$, (enačba (3.10)):

$$\begin{aligned} \mathbf{Z}_o &= \mathbf{Z}_z \\ \mathbf{S}_o &= (x_{n_1}, y_{n_2}) \end{aligned} \tag{3.10}$$

3.5 Detekcija krivulje

Rezultat segmentacije je binarna matrika A, kjer je vrednost elementa a_{ij} (kjer je $i \in [1, M]$ in $j \in [1, N]$ pri velikosti slike $M \times N$ slikovnih elementov) enaka 1 v primeru, da gre za slikovni element na krivulji in 0, če je to slikovni element ozadja. Zaradi debeline zaznane krivulje in morebitnih madežev je v stolpcu j ponavadi več kot en slikovni element z vrednostjo 1. Namen detekcije krivulje je iz segmentirane slike pridobiti natančen potek le te – samo en slikovni element na stolpec, torej seznam točk (i, j) kjer vrednosti i zavzemajo vse vrednosti med začetkom in koncem področja zanimanja, vrednost j pa predstavlja količino padavin v posodi v trenutku i. Postopek mora biti dovolj robusten, da ga ne zmotijo packe, razmazana krivulja in morebitne prekinitve krivulje. Predstavljeni sta dve metodi, ki v sodelovanju lepo detektirata krivuljo.



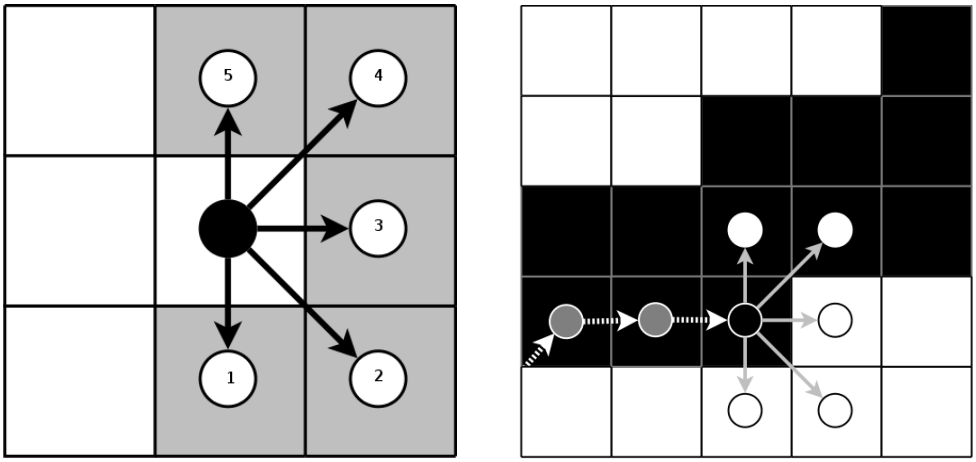
Slika 3.8: Postopek detekcije krivulje.

Detekcija krivulje je implementirana kot kombinacija dveh pristopov, ki se med seboj izmenjujeta (slika 3.8):

- **Lokalno sledenje roba krivulje:**
  hitro in natančno določi potek krivulje tam, kjer je ta povezana. Možne so napake v primeru ramazane krivulje.

- **Globalno dvo-stopenjsko drseče povprečenje:**
  računa povprečno vrednost ordinate preko več sosednjih stolpcev. Je mnogo manj dovzetno za nepravilnosti, vendar tudi manj natančno določa krivuljo.

3.5.1 Lokalno sledenje roba krivulje



(a) Vrstni red obiskovanja. (b) Konkreten primer sledenja.

Slika 3.9: Izbira naslednjega koraka pri sledenju roba.

Metoda sledi spodnjemu robu krivulje. Izbira slikovne elemente v zaporedju: dol, dol-desno, desno, gor-desno, gor – kot to prikazuje slika 3.9. S takšnim zaporedjem izbire korakov zagotovimo, da so rezultirajoče točke vedno na spodnjem robu krivulje. Uporabljeno je sestopanje: v primeru, da ne more naprej, sestopi za maksimalno n slikovnih elementov. Če kljub sestopanju ne pride do napredka pa se postopek ustavi. Delo prepusti koraku povprečenja, ki za osrednji stolpec uporabi naslednji, še neobiskani stolpec. V tem primeru je uporabljena vrednost $n = 10$, v redu pa deluje z n med 5 in 20. Izbrana vrednost n je določena empirično. Postopek prikazuje algoritem 3.2.

```
walkEdge(startPoint, input, backtrackLen){
  // število moznih naslednikov
  BRANCH_FACTOR = 5
  // shranjuje izbrane smeri zadnjih backtrackLen korakov
  queue = []
  output = []       // izhod
  visited = []      // hramba neprimernih točk
  queue.put(1)      // začetna smer je navzdol
  output.push_back(startPoint) // začetna točka
  while not queue.empty() do
    while step() do
      // nič
    if not backtrack() then
      // tudi sestopanje ne pomaga več
      break
  return output
}

step(){
  direction = queue.back()
  // zadnja točka - iščemo naslednjo
  start = output.back()
  // dokler ne najdemo smeri do sosednjega polja,
  // ki še ni bilo obiskano in pripada krivulji
  while korak v smer direction ni mogoc do
    direction++
  if direction >= BRANCH_FACTOR then
    return false
  nextPoint = start.neighborInDirection(direction)
  queue.back() = direction
  if queue.size() > backtrackLen then
    queue.pop_back()
  output.push_back(nextPoint)
  return true
}

backtrack(){
  while not queue.empty() do
    // če je mogoč se kakšen korak iz trenutne točke
    // - poizkusi naslednjega
    if queue.back() < BRANCH_FACTOR then
      queue.back()++
      return true
    else
      // vse možnosti te točke izčrpane
      // - ne obišči je več
      visited.at(points.back()) = true
      queue.pop_back()
      output.pop_back()
  return false
}
```

Algoritem 3.2: Sledenje roba krivulje v psevdokodi.

3.5.2 Globalno dvo-stopenjsko drseče povprečenje

Potek krivulje dobro določa baricenter detektiranih slikovnih elementov, če so le dovolj na gosto razporejeni okrog prave vrednosti. Vendar je kljub temu potrebnih le malo napačno detektiranih slikovnih elementov, ki se nahajajo daleč od poteka krivulje, da dobimo veliko napako. Da se temu izognemo, uporabimo robustnejšo metodo, ki deluje v dveh korakih.

Za vsak stolpec i v prvem koraku izračunamo začetno vrednost $\bar{y}_i$ kot drseče povprečje (angl. moving average) y vrednosti detektiranih slikovnih elementov preko več sosednjih stolpcev:

$$\bar{y}_i = \frac{1}{\bar{N}_i} \sum_{j=1}^{M} \sum_{k=i-\lfloor \frac{W}{2} \rfloor}^{i+\lfloor \frac{W}{2} \rfloor} a_{jk} * j, \tag{3.11}$$

kjer je

$$\bar{N}_i = \sum_{j=1}^{M} \sum_{k=i-\lfloor \frac{W}{2} \rfloor}^{i+\lfloor \frac{W}{2} \rfloor} a_{jk}. \tag{3.12}$$

$\bar{N}_i$ je število vseh detektiranih slikovnih elementov v oknu lihe širine W s središčem v i-tem stolpcu. V primeru, da je število $\bar{N}_i = 0$, je vrednost $\bar{y}_i$ označena kot neznana. Z upoštevanjem več sosednjih stolpcev zmanjšamo vpliv nepravilno detektiranih slikovnih elementov na ocenjeno vrednost $\bar{y}_i$. Večja ko je širina okna W, manjši je vpliv madežev na detekcijo krivulje. Hkrati pa se s povečevanjem širine okna W povečuje tudi efekt glajenja in s tem zmanjšuje natančnost. Kot kompromis je uporabljena vrednost $N = 5$, kar je približno enako debelini krivulje.

Kljub vsemu lahko madeži povprečje odpeljejo daleč od prave vrednosti krivulje. Zato v drugem koraku iz prvega koraka pridobljeno vrednost uporabimo le kot začetni približek, ki določa središče novega, manjšega okna enake širine. Namen drugega koraka je čim tesneje zaobjeti detektirane slikovne elemente okna. Končni približek je nato določen kot:

$$y_i = \frac{1}{N_i} \sum_{j=\bar{y}_i-\lfloor \frac{H_i}{2} \rfloor}^{\bar{y}_i+\lfloor \frac{H_i}{2} \rfloor} \sum_{k=i-\lfloor \frac{W}{2} \rfloor}^{i+\lfloor \frac{W}{2} \rfloor} a_{jk} * j, \tag{3.13}$$

kjer je N_i število vseh detektiranih slikovnih elementov v oknu s središčem v točki $(i, \bar{y}_i)$ širine W ter višine H_i. Višina H_i je odvisna od števila detektiranih slikovnih elementov v začetnem oknu in se izračuna po enačbi (3.14).

$$H_i = \max(2 * W, \frac{\bar{N}_i}{W}) \tag{3.14}$$

Ko imamo določen približek, ki naj bi določal krivuljo v stolpcu i, lahko, kot $N_i/\bar{N}_i$, izračunamo koeficient zaupanja, ki nam pove, koliko lahko temu približku zaupamo. Obnašanje programa glede na vrednost koeficienta zaupanja e_i, izračunanega za i-ti stolpec, je:

- $e_i \leq 0$ 6: detektirana vrednost stolpca i je neznana. Nadaljuje se postopek povprečenja. Osrednji stolpec se pomakne na $i + 1$.
- $e_i \geq 0$ 8 ter slikovni element (i, y_i) je označen kot detektiran: detektirana vrednost stolpca i je enaka y_i. Točka (i, y_i) se doda na konec seznama. Povprečenje prepusti delo metodi sledenja roba krivulje, ki za začetno točko uporabi (i, y_i).
- $e_i > 0$ 6: detektirana vrednost stolpca i je enaka y_i. Točka (i, y_i) se doda na konec seznama. Nadaljuje se postopek povprečenja, kjer se osrednji stolpec pomakne na $i + 1$.

Z uporabo dveh korakov ter koeficienta zaupanja v primeru zmerne količine šuma (madežev) lahko natančno določimo lego krivulje. V primeru velikih madežev pa kljub vsemu lahko pride do napačne detekcije.

Celoten postopek, ki je prikazan tudi grafično na sliki 3.8, je sledeč:

- Poizkusimo poiskati začetek krivulje. Iščemo ga v področju slike O, ki je opisano ter definirano z enačbo (3.10) v razdelku Kalibracija. To je prva slikovna točka tega področja iz katere lahko z metodo sledenja roba krivulje naredimo vsaj 100 korakov. Število korakov je določeno empirično ter poskrbi, da nas ne zavedejo manjši madeži. V primeru, da takšno točko najdemo, nadaljujemo s sledenjem roba, dokler je to mogoče, sicer začnemo detekcijo z metodo povprečenja, ki za začetni stolpec vzame prvi stolpec področja zanimanja.
- Izmenjujeta se korak sledenja krivulje ter korak povprečenja, dokler ne obiščemo vseh slikovnih stolpcev segmentirane slike. Pravila, po katerih se metodi izmenjujeta, so opisana zgoraj ter grafično predstavljena tudi na sliki 3.8.

V končnem koraku detekcije je potrebno dobljeno zaporedje še sploščiti, saj se kot posledica sledenja roba v posameznem stolpcu i lahko nahaja več

točk. V takšnem primeru te točke nadomestimo z eno samo. Vrednost y_i je v tem primeru enaka povprečni vrednosti takšnih točk.

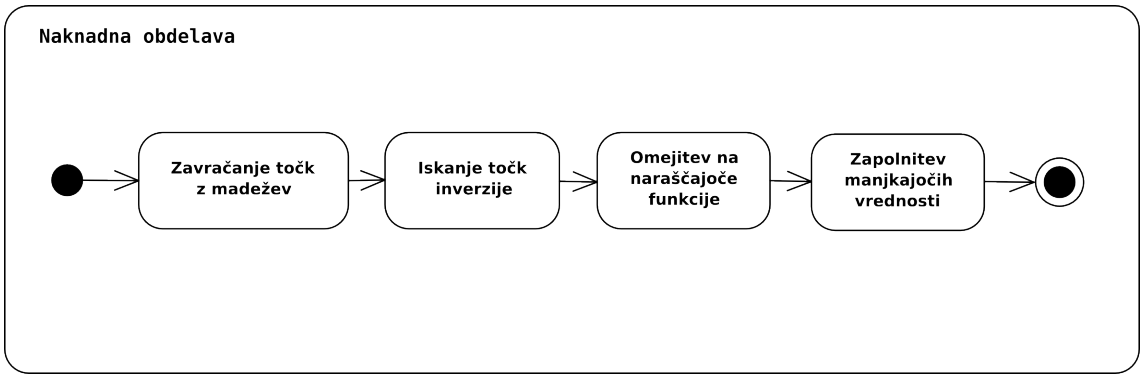
Rezultat detekcije krivulje je zaporedje točk $\mathbf{a}_i = (x_i, y_i)$ z največ eno točko na stolpec vhodne slike. Takšno zaporedje je primerno za prikaz in oceno rezultata koraka detekcije.

3.6 Naknadna obdelava

Rezultat koraka detekcije krivulje je seznam $\mathbf{a}_i = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, .., \mathbf{a}_n)$, kjer vsak $\mathbf{a}_i$ predstavlja točko detektiranega poteka krivulje. Med točkami so mogoči presledki, torej stolpci, kjer potek krivulje ni definiran.

Namen koraka naknadne obdelave je:

- odpraviti morebitne napake koraka detekcije
- pridobiti zaporedje brez presledkov – zapolniti manjkajoče vrednosti
- pridobiti zaporedje, ki je odsekoma naraščajoče.



Slika 3.10: Funkcije naknadne obdelave.

Korak je sestavljen iz več funkcij, ki postopoma preoblikujejo vhodno zaporedje v takšno, ki izpolnjuje naštete zahteve. Te funkcije obdelujejo le v koraku detekcije pridobljeno zaporedje točk (ne procesirajo same slike). Prikazane so na sliki 3.10.

3.6.1 Zavračanje točk z madežev

Poizkusimo odkriti točke, ki so posledica madežev. Prepoznamo jih po hitrih skokih zaporedja. Pri tem je treba paziti, da po nesreči ne odstranimo napačnih točk – točk krivulje. Določiti je potrebno takšne pogoje, da se to ne bo zgodilo. To naredimo tako:

1. Poiščemo točko, kjer zaporedje strmo skoči. Poiščemo takšni dve zaporedni točki $\mathbf{a}_i = (x_1, y_1)$ in $\mathbf{a}_{i+1} = (x_2, y_2)$, da velja:

$$|y_1 - y_2| > \frac{\text{višina področja zanimanja}}{3}. \tag{3.15}$$

2. Poizkusimo poiskati točko, kjer zaporedje strmo skoči v nasprotno smer kot prej. Obiščemo točke $\mathbf{a}_j$, kjer gre j od $i+1$ do $i+100$. Poizkušamo najti takšni zaporedni točki $\mathbf{a}_j = (x_3, y_3)$ in $\mathbf{a}_{j+1} = (x_4, y_4)$, da velja:

$$|y_3 - y_4| > \frac{\text{višina področja zanimanja}}{4}, \tag{3.16}$$

   smer skoka pa je obratna smeri prvega. V primeru da takšni točki najdemo, iz zaporedja zbrišemo točke med $\mathbf{a}_i$ in $\mathbf{a}_{j+1}$.

3. Postopek ponavljamo toliko časa, dokler je mogoče najti v korakih 1 in 2 opisane skoke.

Namen tega koraka ni odstraniti vseh točk, ki pripadajo madežem, temveč le tiste, ki bi lahko v naslednjem koraku rezultirale v napačno detektirani inverziji. Preostale točke madežev se nato kasneje eliminira v koraku omejitve na rastoče funkcije. Vrednost 100, ki smo jo uporabili v koraku 2 in določa maksimalno razdaljo med skokoma in s tem maksimalno velikost madeža, je določena s poskušanjem nad učno množico slik. V primeru spremembe ločljivosti vhodne slike je to vrednost potrebno ustrezno prilagoditi. Pogoja, ki definirata skoke, sta določena empirično in dovolj strogo, da ne izgubimo pravih točk krivulje.

3.6.2 Iskanje točk inverzije

Poiščemo točke, kjer se gibanje pisala strmo obrne navzdol – merilna posoda se izprazni. Izpraznjenje posode je v zaporedju točk opazno kot velik skok z zgornje do spodnje meje področja zanimanja. Točke inverzij so definirane kot točke zaporedja, ki predstavljajo lokalni minimum in se nahajajo v zgornji

tretjini področja zanimanja, ki jim v naslednjih k členih zaporedja sledi točka, katere razdalja po ordinatni osi je večja od polovice višine področja zanimanja. Izbrana vrednost $k = 20$ je določena empirično in je odvisna od ločljivosti vhodne slike. Izmed teh točk izberemo tisto z maksimalno vrednostjo ordinatne osi. Podzaporedje med točko inverzije in maksimumom nima vpliva na nadaljnje izračune, zato ga iz zaporedja izločimo. Točke inverzij shranimo.

3.6.3 Omejitev na naraščajoče funkcije

Ko so znane točke inverzij krivulje, za vsak odsek zaporedja točk, ki je bodisi med dvema takšnima točkama, bodisi vsebuje začetno ali končno točko, vemo, da bi morale te točke predstavljati naraščajoče zaporedje. Vendar pa rezultat koraka detekcije krivulje nima nujno take oblike. Ker detektirana krivulja nima povsem gladkih robov, lahko pride v odsekih brez padavin do rahlih odstopanj od popolnoma vodoravne premice. Lahko se zgodi tudi, da nam v koraku zavračanja točk z madežev vseh takšnih točk ni uspelo odstraniti. Vsa ta odstopanja predstavljajo neke vrste šum. Kljub vsemu temu pa lahko trdimo, da se večina točk nahaja na odsekoma naraščajoči krivulji. Zaupamo večini: večina točk je pravilno detektiranih, večina točk predstavlja naraščajoče zaporedje, torej najdaljše naraščajoče podzaporedje dobro aproksimira pravi potek krivulje.

Na vsakem odseku, ki jih ločujejo točke inverzij določimo optimalni potek krivulje z uporabo algoritma za iskanje najdaljšega naraščajočega podzaporedja 3.3.

Najdaljše naraščajoče podzaporedje

Najdaljše naraščajoče podzaporedje (v nadaljevanju NNP) [9, 4] danega zaporedja števil $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \ldots, \mathbf{a}_n)$ je takšno zaporedje $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \ldots, \mathbf{z}_k)$, ki ga lahko predstavimo z elementi zaporedja $\mathbf{A}$ kot $\mathbf{Z} = (\mathbf{a}_{j_1}, \mathbf{a}_{j_2}, \mathbf{a}_{j_3}, \ldots, \mathbf{a}_{j_k})$ ter velja

- zaporedje $\mathbf{Z}$ je podzaporedje zaporedja $\mathbf{A}$:
  $1 \leq j_1 < j_2 \ldots < j_k \leq n$
- zaporedje $\mathbf{Z}$ je naraščajoče:
  $\forall \mathbf{z}_i \in \mathbf{Z} : \mathbf{z}_i \leq \mathbf{z}_{i+1}$
- nobeno drugo naraščajoče podzaporedje zaporedja $\mathbf{A}$ ni daljše kot $\mathbf{Z}$:
  $\{\nexists \mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_l)$: ($\mathbf{X}$ je NP zaporedja $\mathbf{A}$) $\} \wedge \{l > k\}$.

Algoritem za iskanje najdaljšega naraščajočega podzaporedja

Algoritem za iskanje NNP [4, 9] je lep primer dinamičnega programiranja. Kot vhod sprejme poljubno zaporedje števil $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \ldots, \mathbf{a}_n)$ ter vrne NNP tega zaporedja. Definirajmo NNP(i) kot najdaljše naraščajoče podzaporedje zaporedja $\mathbf{A}$, ki se konča z elementom $\mathbf{a}_i$. Elemente zaporedja $\mathbf{A}$ obiskujemo enega za drugim. Pri tem v tabelah L in *prev* hranimo podatke, ki omogočijo izračun ter rekonstrukcijo NNP. Element $L[i]$ hrani dolžino NNP(i) ter je določen kot

$$L[i] = \begin{cases} 1 & ; \text{če } i = 1 \\ 1 + max_{j<i}\{L[j] : a_j \leq a_i\} & ; \text{sicer} \end{cases}, \tag{3.17}$$

kar predstavlja srž algoritma. Najdaljše podzaporedje NNP(i) je z elementom $\mathbf{a}_i$ podaljšano najdaljše zaporedje izmed NNP(1), NNP(2), ..., NNP($i-1$). NNP celotnega zaporedja $\mathbf{A}$ je tako enako NNP(n), kjer je n enak dolžini zaporedja $\mathbf{A}$.

Ob izteku algoritma maksimalna vrednost tabele L predstavlja dolžino NNP. Ker pa nas zanima tudi NNP samo, potrebujemo še spremenljivko *end*, ki hrani indeks konca NNP in tabelo predhodnikov *prev*. V tabeli *prev* se shranjujejo indeksi elementov tabele $\mathbf{A}$ tako, da

$$prev[i] = \begin{cases} 0 & ; \text{element } \mathbf{a}_i \text{ nima predhodnikov} \\ k & ; \text{element } \mathbf{a}_k \text{ je predhodnik elementa } \mathbf{a}_i \text{ v NNP}(i) \end{cases}. \tag{3.18}$$

Na tem mestu je potrebno pripomniti, da se uporabljeni pojem "naraščajoč" nanaša na potek krivulje po traku pluviografa. Dejansko pa gre, zaradi koordinatnega sistema slike, katerega izhodišče je v skrajnem levem zgornjem kotu slike, za iskanje najdaljšega padajočega zaporedja, kar velja tudi za algoritem 3.3. Edina razlika je, da pri iskanju najdaljšega padajočega podzaporedja v (3.3, vrstica 14), zamenjamo (**if** A[j] <= A[i] **then**) z (**if** A[j] >= A[i] **then**).

```
1  NNP(A){
2    n = A.length
3    result = []
4    len = 1
5    end = 1
6    L[1] = 1
7    prev[1] = 0
8    // čez vse elemente zaporedja A
9    for i = 1 to n do
10     L[i] = 1
11     prev[i] = 0
12     // čez vse elemente do i
13     for j = 1 to i - 1 do
14       if A[j] <= A[i] then
15         if L[i] < 1 + L[j] then
16           // NNP[j] lahko podaljšamo z A[i]
17           L[i] = L[j] + 1
18           prev[i] = j
19     if len < L[i] then
20       len = L[i]
21       end = i
22   // rekonstruiramo NNP
23   while prev[end] != 0 do
24     result.push_front(A[end])
25     end = prev[end]
26   return result
27 }
```

Algoritem 3.3: Algoritem za iskanje najdaljšega naraščajočega podzaporedja.

3.6.4 Zapolnitev manjkajočih vrednosti

Trenutno zaporedje lahko vsebuje presledke. Ti so lahko posledica koraka detekcije, kjer v določenem stolpcu korak povprečenja ni dobil zadostnega koeficienta zaupanja. Lahko pa so nastali kasneje v vseh predhodnjih korakih naknadne obdelave: zavračanje točk z madežev, iskanje točk inverzije ter omejitev na naraščajoče funkcije. Manjkajoče vrednosti se določijo z linearno interpolacijo. Posebna oblika presledkov se pojavi ob točkah inverzije, torej na mejah naraščajočih zaporedij. V tem primeru uporabimo poznavanje merilnega instrumenta. Razpon med ekstremoma inverzije je namreč vedno enak višini področja zanimanja.

V primeru presledka med naraščajočima točkama $\mathbf{a}_i = (x_1, y_1)$ in $\mathbf{a}_{i+1} = (x_2, y_2)$, za kateri velja $(x_1 + 1 < x_2)$, med njiju vstavimo manjkajoče točke

$\mathbf{a}_{j_1}, \mathbf{a}_{j_2}, \mathbf{a}_{j_3}, \ldots, \mathbf{a}_{j_t}$. V prazen stolpec l ($x_1 < l < x_2$) vstavimo točko $\mathbf{a}_j = (l, y)$, kjer y izračunamo po enačbi (3.19).

$$y = y_1 + \frac{(l - x_1) * (y_2 - y_1)}{(x_2 - x_1)} \tag{3.19}$$

V primeru interpolacije presledkov pri inverzijah postopamo enako, le da namesto $\mathbf{a}_{i+1} = (x_2, y_2)$ (desna meja presledka) uporabimo fiktivno točko $\mathbf{a}_f = (x_2, y_f)$. Gre torej za interpolacijo med $\mathbf{a}_i$ in $\mathbf{a}_f$, kjer je $\mathbf{y}_f$ je določen kot:

$$y_f = y_2 - \text{višina področja zanimanja.} \tag{3.20}$$

To je položaj točke, ki bi ga imela točka $\mathbf{a}_{i+1}$ v primeru, da se posoda ne bi praznila – ne bi bilo inverzij. Tako podaljšamo naraščajoče zaporedje, ki se konča z elementom $\mathbf{a}_i$ do točke $\mathbf{a}_f$. Pozorni moramo biti na indekse, saj moramo upoštevati, da je izhodišče koordinatnega sistema v skrajnem levem zgornjem kotu slike, kar upoštevata tudi enačbi (3.19) in (3.20).

Rezultat koraka naknadne obdelave je odsekoma naraščajoče zaporedje točk (koordinat slikovnih elementov) z natanko eno točko na stolpec področja zanimanja slike.

3.7 Izpis rezultatov

Dobljeno odsekoma naraščajoče zaporedje (dolžine n) pretvorimo v kumulativno zaporedje razlik ordinatnih vrednosti med točkami (enačba (3.21)). Razlika ordinatnih vrednosti dveh zaporednih elementov zaporedja predstavlja količino padavin v času določenim s širino slikovnega elementa. Tako lahko za katerikoli točki zaporedja $\mathbf{a}_i = (x_i, y_i)$ in $\mathbf{a}_j = (x_j, y_j)$ $(i < j)$ za časovni interval $[i, j]$ izračunamo količino padavin izraženo v koordinatah točk kot $(\mathrm{cumm}_j - \mathrm{cumm}_i)$. Vendar nam količina padavin izražena v koordinatah slikovnih elementov slike ne pove prav veliko. Zdaj lahko s pomočjo mer področja zanimanja zaporedje točk slike pretvorimo v nam zanimivo količino – intenziteto padavin.

$$\begin{aligned} &\mathrm{cumm}_0 = 0 \\ &\mathrm{cumm}_i = \mathrm{cumm}_{i-1} + (y_{i-1} - y_i) \qquad 1 \geq i \geq n \end{aligned} \tag{3.21}$$

Pri enačbi (3.21) je treba ponovno opozoriti na to, da je izhodišče koordinatnega sistema slike v skrajnem zgornjem levem kotu. Posledica tega je, da je vrednost y_i manjša od vrednosti y_{i-1}. Upoštevati pa je potrebno tudi meje naraščajočih zaporedij, Tako lahko za poljuben časovni interval $[t, t+m]$ $(t, t+m \in [0, 1440\ \mathrm{min}])$ izračunamo intenziteto padavin s pomočjo enačbe:

$$\mathrm{intenziteta}_{[\mathrm{t,t+m}]} = \frac{(\mathrm{cumm}_{\mathrm{k+l}} - \mathrm{cumm}_{\mathrm{k}}) * (10\ \mathrm{mm})}{(\text{višina področja zanimanja})}, \tag{3.22}$$

kjer sta k in $k+l$ indeksa zaporedja cumm, ki ustrezata časovnima trenutkoma t in $t+m$ (enačba (3.23)). Razmerje (10 mm)/(višina področja zanimanja) (razmerje med količino padavin v posodi ob praznjenju posode ter številom slikovnih elementov v stolpcu področja zanimanja) določa padavinsko ločljivost – najmanjšo intenziteto padavin, ki jo lahko zabeležimo.

$$\begin{aligned} k &= \lfloor \frac{(t * \text{širina področja zanimanja})}{1440\ \mathrm{min}} \rfloor \\ l &= \lfloor \frac{(m * \text{širina področja zanimanja})}{1440\ \mathrm{min}} \rfloor \end{aligned} \tag{3.23}$$

Razmerje 1440 min/(širina področja zanimanja) določa časovno ločljivost – najmanjši možni časovni interval m. Konstanta 1440 je število minut v 24 urah.

Poglavje 4

Rezultati

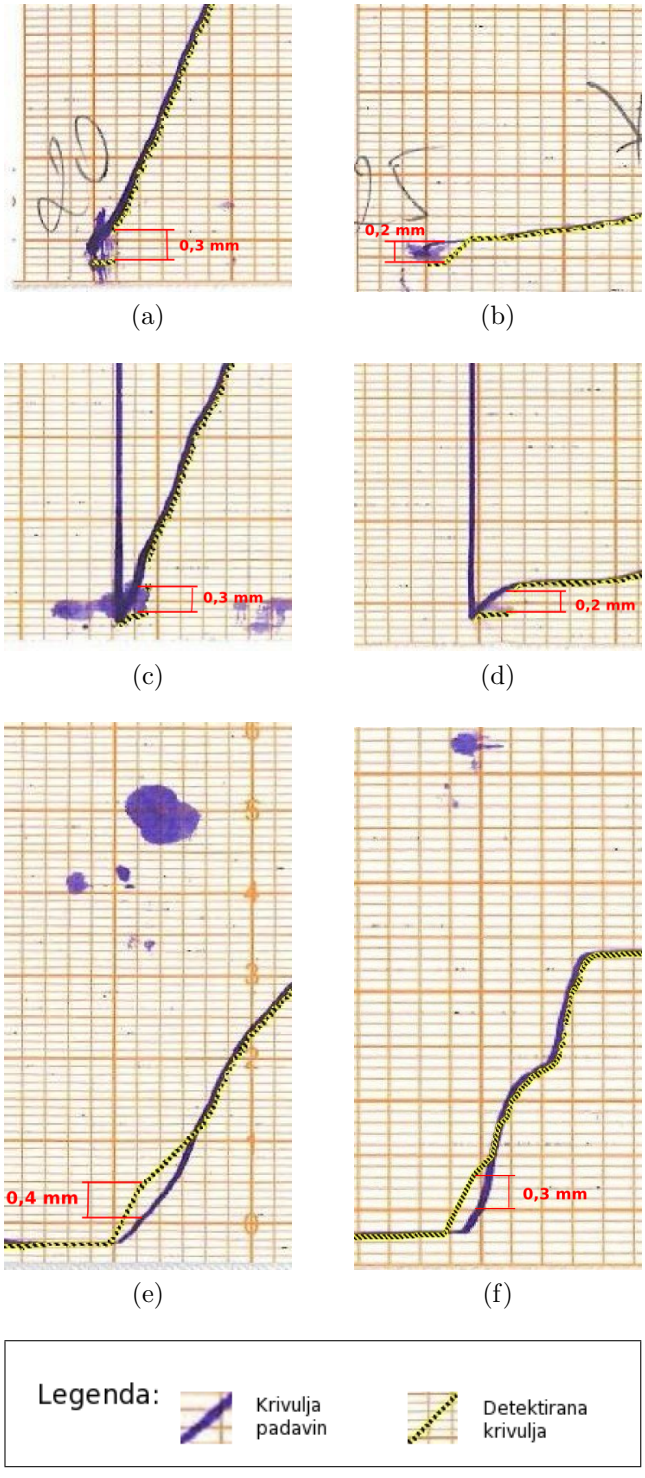
Algoritem je bil testiran nad testno množico slik, ki ni bila uporabljena za izdelavo algoritma. Testna množica sestoji iz 58 slik pluviografskih trakov z meteorološke postaje Podkraj za leto 2006. Rezultati so ocenjeni kvalitativno, glede na vnaprej definirane razrede odstopanj, nato pa še kvantitavno.

4.1 Kvalitativna ocena

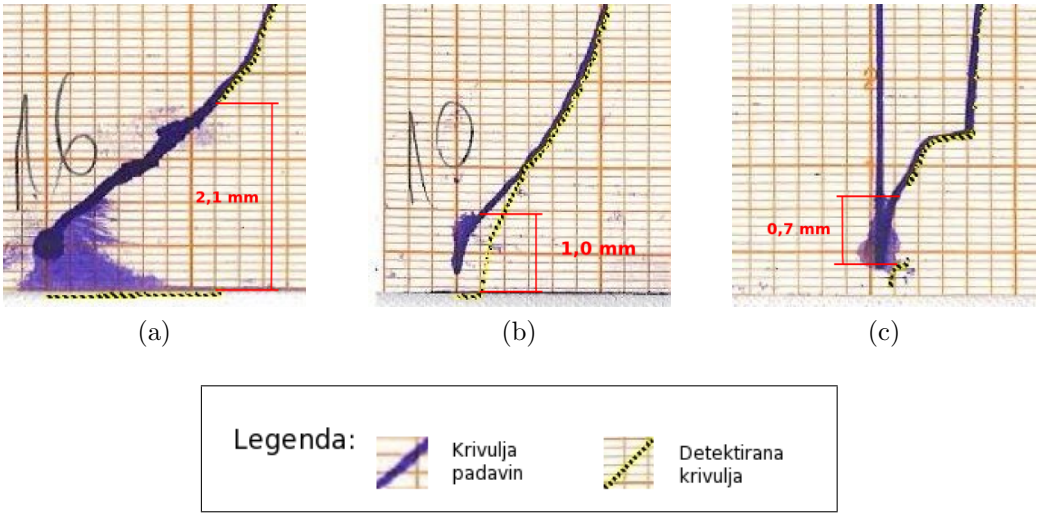
V tem poglavju je ocenjen rezultat koraka detekcije krivulje. Po ogledu detektirane krivulje se rezultatu pripiše natančnost/pravilnost glede na vnaprej določene razrede. Na sliki, ki nam jo prikaže program, primerjamo potek krivulje z detektiranim potekom, ki je prikazan na isti sliki, in poiščemo stolpec v katerem je vertikalno odstopanje največje – to odstopanje nato določi razred napake. Uporabljeno merilo so mm, ki so označeni na traku ter omogočajo enostavno odčitavanje napake. To merilo nima neposredne povezave z intenziteto padavin, manjka mu namreč časovna komponenta (interval). Vse slike ne ustrezajo pogojem določenim v začetku poglavja 3. Kljub temu pa se postopek v večini primerov izkaže kot dober, kar je razvidno v tabeli 4.1.

Razredi:

- **Razred 0**: Ni opaznih napak ali odstopanj od poteka krivulje ali pa so ta odstopanja manjša od 0,2 mm (želimo čim več elementov v tem razredu).
- **Razred 1**: Prišlo je do manjših odstopanj od krivulje. Največje opaženo odstopanje od pravilne lege krivulje je večje od 0,2 mm in manjše od 0,5 mm (slika 4.1).



Slika 4.1: Primeri napak, kjer je največje odstopanje med 0,2 in 0,5 mm (razred 1).



(a) (b) (c)

Slika 4.2: Primeri napak, kjer je največje odstopanje večje od 0,5 mm ter manjše od 10 mm (razred 2).

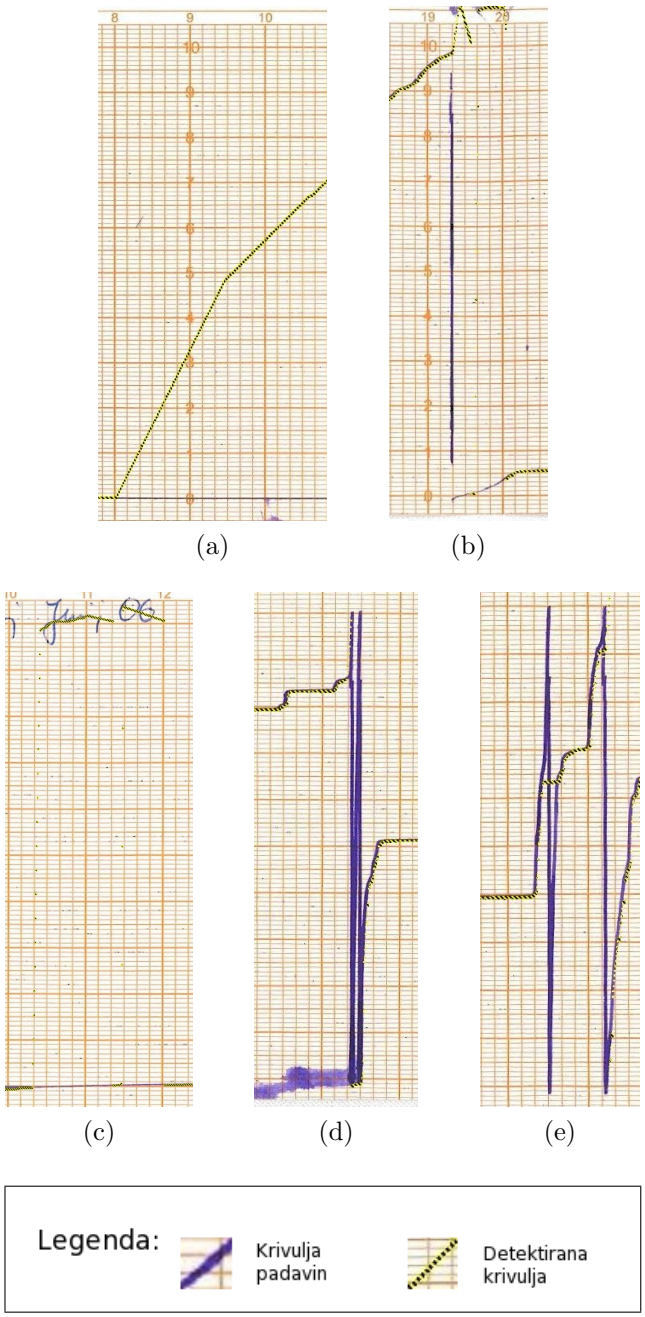
- **Razred 2**: Prišlo je do večjih odstopanj od krivulje. Največje opaženo odstopanje od pravilne lege krivulje je večje od 0,5 mm in manjše od 10 mm (slika 4.2).
- **Razred 3**: Prišlo je do napak, ki vodijo v napačen izračun celodnevne količine padavin za več kot 10 mm (npr. napačno detektirana inverzija) (slika 4.3).

Vsi elementi razredov 1, 2 in 3 so prikazani na slikah 4.1, 4.2 in 4.3

Razred 0	Razred 1	Razred 2	Razred 3
44	6	3	5

Tabela 4.1: Rezultati kvalitativnega testiranja.

Ob izbranem časovnem intervalu vplivajo na rezultat le napake, ki se raztezajo preko meja dveh ali več intervalov. Odstopanja omejena na notranjost intervala tako ne vplivajo na rezultat. Intenziteta padavin elementov razreda 1 in 2, katerih napake so omejene na notranjost intervalov, se kljub temu pravilno izračuna. Opaziti je, da se v robnih intervalih (začetek in konec krivulje)



Slika 4.3: Primeri napak, kjer je odstopanje celodnevnih padavin večje od 10 mm (razred 3).

napake pogosteje pojavljajo (zaradi večje verjetnosti madežev in nepoznavanja nadaljnjega poteka krivulje). Razlogi za napake vseh primerov iz razredov 1 in 2 so madeži. Ti vplivajo na korak detekcije krivulje (poglavje 3.5):

- sledenje roba krivulje lahko zaide: slike 4.1 (a), (c), (d) ter vse slike 4.2.
- v primeru, da je na odseku, ki ima madeže aktivno povprečenje, lahko postopek bodisi najde napačne točke, bodisi označi vrednosti v stolpcih za neznane. Do neke mere se takšne napake lahko popravijo v koraku naknadne obdelave, v primeru velike količine madežev pa lahko kljub vsemu pride do odstopanj od krivulje, kot je opaziti na slikah 4.1 (e) in (f).

Za izboljšanje rezultatov bi lahko v koraku segmentacije implementirali obliko lokalno prilagojenega upragovanja. V koraku naknadne obdelave - zapolnitev manjkajočih vrednosti pa bi bilo mogoče primerno pred interpolacijo še enkrat poizkusiti poiskati potek krivulje med robnima točkama.

V primerih iz razreda 3, ki so prikazani na sliki 4.3, pride do hujših napak, kot so napačno določene inverzije krivulje: (a), (b) in (c). V primerih slik (d) in (e) pa je bila inverzija spregledana – ni bila zaznana.

V primeru slik 4.3 (a) in (c) je do napake prišlo zaradi zelo slabo vidne krivulje, ki je bila naknadno popravljena s črnim pisalom in je posledično na segmentirani sliki praktično neopazna. Slabše vidna krivulja po inverziji ter madež sta botrovala napaki s slike 4.3 (b). Na sliki (d) je opaziti dve inverziji v zelo kratkem časovnem obdobju, kar je algoritem zaznal kot le eno inverzijo. K tej napaki je veliko prispeval tudi madež v spodnjem delu. Pri sliki (e) gre prav tako za spregledano inverzijo, ki pa bi jo bilo zelo verjetno mogoče odpraviti s pravilno nastavitvijo parametrov algoritma. Sliki (a) in (c) ne ustrezata omejitvam, ki so naštete na začetku poglavja 3.

4.2 Kvantitativna ocena

Primerjali smo vrednosti, pridobljene z v tem delu opisanim algoritmom, z vrednostmi, ki jih hrani Agencija Republike Slovenije za okolje.

Pri kvantitativni oceni napake nismo upoštevali primerov, ki pripadajo razredu 3 (niso reprezentativni, saj bi ti primeri nujno potrebovali ročno popravljanje) ter primerov slik pluviografskih trakov, katerih podatki za primerjavo niso primerni (dnevi z več kot 5 cm zapadlega snega ter dnevi, ki so beležili le taljenje snega). V primeru snega je postopek merjenja količine padavin kompleksnejši – potrebno je upoštevati tako količino padavin izmerjeno na traku,

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.

kot tudi zapadli sneg, ki pa se ne stopi takoj. Tako podatki zabeleženi na trakovih ne odražajo pravega časovnega poteka padavin in se lahko kvantitativno močno razlikujejo od dejansko zabeleženih. Pri podatkih z urno, polurno ter 5 minutno ločljivostjo so vrednosti nekaterih intervalov manjkale, kar je dodatno zmanjšalo število možnih primerjav. Rezultati so predstavljeni v tabeli 4.2. Napako smo ocenjevali z merami, ki se tipično uporabljajo za ocenjevanje natančnosti/primernosti regresijske krivulje. Tu sta meri MAE in RMAE uporabljeni zato, ker lahko tudi na ta problem gledamo kot neke vrste iskanje funkcije, ki se čim bolje prilega podanim točkam. Meri uporabljeni za oceno napake sta:

- Srednja absolutna napaka (MAE) [6]:

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

  MAE izražen v odstotkih pa je izračunan kot:

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

  Mera MAE nam pokaže, za koliko mm se izračunani podatki povprečno razlikujejo od uradnih podatkov. Torej je povprečna absolutna razlika med primerjanimi podatki. MAEP prikazuje napako MAE v odstotkih – za koliko odstotkov se izračunana vrednost povprečno razlikuje od željene vrednosti.

- Relativna srednja absolutna napaka (RMAE) [6]:

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

Mera RMAE nam prikaže odstopanje izračunanih podatkov od uradnih relativno – glede na razpon možnih vrednosti funkcije. Vrednost RMAE je v tem primeru vedno med 0 in 1, kjer 0 pomeni popolno ujemanje podatkov.

V tem primeru N predstavlja število intervalov, ki smo jih primerjali, $f(i)$ je vrednost zabeležena na Agenciji Republike Slovenije za okolje, $\hat{f}(i)$ pa izračunano vrednost (za i-ti interval). Povprečno vrednost $\bar{f}(i)$, uporabljeno v enačbi (4.3) pa smo izračunali po enačbi:

$$\bar{f} = \frac{1}{N} \sum_{i=1}^{N} f(i). \tag{4.4}$$

Iz tabele (4.2) je razvidno, da je relativno odstopanje (RMAE) razmeroma majhno in se povečuje s krajšanjem časovnega intervala. Pri analizi dobljenih rezultatov je dobro vedeti, da se ti lahko razlikujejo tudi zaradi človeškega faktorja – trak je lahko zamenjan kasneje kot točno ob 7:00 ali pa je nenatančno vstavljen. Za razrešitev takšnih težav je potrebno pomakniti časovno skalo tako, da se ujema z realnim stanjem. Ker podatkov o opravljenih pomikih skale nimamo, lahko v primeru takšnih podatkov pride do odstopanj – napak. Te pa se seveda bolj poznajo, ko je izbrani časovni interval manjši.

Testiranje se je izvajalo na osebnem računalniku z dvojedrnim procesorjem na katerem teče operacijski sistem Linux. Algoritem je zasnovan linearno ter ne izkorišča paralelnega delovanja. Za obdelavo trikratne ponovitve testne množice (3×58 slik) je postopek potreboval 15 minut in 18 sekund. Za povprečno obdelavo posamezne slike traku je torej porabil približno 5,28 sekunde. Za celotno obdelavo pa je temu času potrebno prišteti še čas branja z optičnim čitalnikom ter morebitni čas interaktivnega procesiranja.

Rezultat testiranja na celodnevni (MAE = 0,3844 mm) ter urni (MAE = 0,1350 mm) časovni ločljivosti je zelo dober. V primeru višje časovne ločljivosti pa so odstopanja precej velika (v primeru 5 minutnega intervala celo 42,77 odstotna), kar pa je pričakovano, saj se načina pridobitve podatkov zelo razlikujeta. Medtem, ko algoritem določi točko krivulje v vsakem stolpcu slikovnih elementov slike, so točke pri digitalizaciji z digitalizatorsko tablo izbrane manj na gosto, vmesne točke pa so dobljene z linearno interpolacijo. Tako je potek krivulje, ki ga izračuna algoritem, v primeru pravilne detekcije krivulje natančneje določen. Upoštevati pa je potrebno tudi morebitne zgoraj omenjene premike časovne skale. Kljub temu, da v tem poglavju govorimo o napakah, ta izraz ni povsem na mestu. Gre bolj za primerjavo dveh metod, ki sta obe podvrženi lastnim napakam.

4.3 Pomanjkljivosti ter nadaljnje delo

Najpogostejši razlogi za slab rezultat so:

- Slab zapis krivulje (4.3 (b)).
- Ročno narejeni popravki z barvo, ki ni modra (sliki 4.3 (a) in (c)).
- Velika količina ter velikost madežev, še zlasti na robovih (začetek in konec poteka krivulje) ter ob inverzijah (slike 4.3 (d), 4.2 (a) in (c)).
- V primeru zelo visoke intenzitete padavin, so inverzije preveč blizu skupaj, da bi jih algoritem lahko razločil (slika 4.3 (d)).

Pametno iskanje madežev ter posebna obravnava teh področij v koraku segmentacije bi lahko omilila napake zaradi madežev. Postopek bi lahko v koraku segmentacije izboljšali z implementacijo lokalno prilagojenega upragovanja. Primerno bi bilo uporabiti tudi L^* komponento prostora barv CIELAB, saj brez nje nekako izpustimo del podatkov, ki tudi nosijo določeno količino informacije. Iskanje inverzij bi bilo mogoče izboljšati z upoštevanjem informacije segmentirane slike. Postopek bi lahko poleg opisanega poizkusil tudi na sliki odkriti značilnosti inverzij. Iskanje najdaljšega naraščajočega zaporedja (Poglavje 3.6.3) je v nekaterih primerih, ki sicer skoraj nikoli ne ustrezajo pogojem naštetim na začetku poglavja 3, morda prestrog kriterij. Morda bi se lahko isti učinek dobilo z uporabo nelinearne regresije z eliminacijo od krivulje oddaljenih točk, kot to opisuje [8]. Pri tem bi bilo potrebno omejiti iskanje regresijske krivulje na naraščajoče funkcije. V obzir je potrebno vzeti tudi čas delovanja, saj bi le tega vse naštete izboljšave občutno podaljšale.

Glede na rezultate lahko ocenimo algoritem kot dober, kljub temu pa kot samostojna (avtomatska) aplikacija ni primeren. Doseganje visoke natančnosti ter majhne količine napačnih detekcij je namreč zaradi raznolikosti vseh mogočih dejavnikov težko dosegljivo. Napake je mogoče opaziti, ne pa tudi odpraviti. Algoritem je tako potrebno vključiti v sistem, ki prikaže detektiran potek krivulje in ob pravilni detekciji omogoča potrditev ter shranjevanje rezultatov, v nasprotnem primeru pa tudi ročno popravljanje. Primerna orodja za takšno interaktivno procesiranje bi lahko omogočala:

- brisanje madežev,
- vertikalno premikanje točk detektirane krivulje,
- dodajanje/brisanje točk detektirane krivulje,

- dodajanje/brisanje točk inverzij,
- premik časovne skale (včasih trak ni zamenjan točno ob 7:00).

Sistem bi lahko vključeval tudi korak branja z optičnim čitalnikom. Tako bi lahko poskrbeli, da so nastavitve optičnega čitalnika vedno enake in avtomatično poravnali sliko.

Poglavje 5

Zaključek

V tem delu je predstavljen algoritem za avtomatsko branje padavinskih grafov s papirnatih trakov pluviografov. Omogoča digitalizacijo padavinskega časovnega zaporedja visoke ločljivosti, ki je bistven element mnogih hidroloških in meteoroloških analiz ter študij. Trenutna postopka shranjevanja teh podatkov v digitalno obliko, ročno oziroma s pomočjo digitalizatorske table, sta zamudna ter monotona. S pomočjo v tem delu opisanega postopka bi lahko ta proces občutno pospešili ter morda povečali tudi natančnost tako pridobljenih podatkov. Mogoča pa je tudi obdelava starejših pluviografskih trakov, kar bi omogočilo boljši vpogled v preteklo padavinsko dogajanje – podalo natančnejšo statistično sliko. Algoritem je bil razvit z namenom, da bi:

- pospešil obstoječi postopek,
- čim natančneje določil potek krivulje,
- vrnil razmeroma dobre rezultate tudi v primeru slabo zaznavnega signala padavin (robustnost),
- omogočil ogled dobljenih rezultatov,
- omogočil shranjevanje rezultatov v željeni obliki.

Sestavljen je iz več korakov, ki postopoma pripeljejo do željenega rezultata, če le vhodna slika ustreza razmeroma ohlapnim omejitvam:

- Slika je poravnana. Vse slikovne točke stolpca i ustrezajo istemu časovnemu intervalu.
- Na grafu mora biti le ena krivulja (v primeru dni brez dežja se včasih uporabi isti trak, kar rezultira v več krivuljah).

- Uporabljeno je modro črnilo pisala.

Poleg metod računalniškega vida, se opira tudi na poznavanje mehanskih lastnosti merilnega instrumenta. Učinkovitost algoritma je bila testirana nad neodvisno množico 58 slik pluviografskih trakov. Glede na rezultate lahko ocenimo algoritem kot dober. Ker pa ne dosega 100 odstotne točnosti, kar je zaradi velike raznolikosti mogočih dejavnikov praktično težko izvedljivo, ni primeren kot samostojna (avtomatska) aplikacija. Potrebno ga je vključiti v sistem z grafičnim vmesnikom, ki omogoča ogled detektiranega poteka krivulje padavin ter morebitno ročno popravljanje le tega. Mogoče so mnoge izboljšave, katerih implementacija bi delovanje algoritma lahko izboljšala, hkrati pa bi se povečal tudi čas procesiranja.

Slike

Tabele

Algoritmi

Literatura

[1] Mitja Brilly, Mojca Šraj, *Osnove hidrologije* - 1. izd, Ljubljana: Fakulteta za gradbeništvo in geodezijo, 2005.

[2] Roberto Deidda, Giuseppe Mascaro, Enrico Piga, Giorgio Querzoli, "An automatic system for rainfall signal recognition from tipping bucket gage strip charts", *Journal of Hydrology*, 333, str. 400-412, 2007.

[3] David A. Forsyth, Jean Ponce, *Computer Vision - A Modern Approach*, Prentice Hall, 2002.

[4] Dan Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, New York, Cambridge University Press, poglavje 12.5, 1997.

[5] Bernd Jähne, *Digital Image Processing*, 5th revised and extended edition, Springer, 2002.

[6] Igor Kononenko, *Strojno učenje*, Ljubljana, Fakulteta za računalništvo in informatiko, str. 51-52, 1997.

[7] Ashidi N. Mat-Isa, Yusof M. Mashor, Hayati N. Othman, "Seeded Region Growing Features Extraction Algorithm; Its Potential Use in Improving Screening for Cervical Cancer", *International Journal of The Computer, the Internet and Management*, 13(1), str. 61-70, 2005.

[8] Harvey J. Motulsky, Ronald E. Brown, "Detecting outliers when fitting data with nonlinear regression – a new method based on robust nonlinear regression and the false discovery rate", BMC Bioinformatics, 2006. (http://www.biomedcentral.com/1471-2105/7/123 (1.3.2009))

[9] Geppino Pucci, Dati e Algoritmi 2, Chapter 5, Dynamic Programming: http://www.dei.unipd.it/~geppo/DA2/DOCS/dynprog.pdf (1.3.2009)

[10] Jože Rakovec, Tomaž Vrhovec, *Osnove meteorologije za naravoslovce in tehnike*, Ljubljana, Društvo matematikov, fizikov in astronomov Slovenije, 1998.

[11] Emanuele Trucco, Alessandro Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

[12] Wikipedia CIELAB:
http://en.wikipedia.org/wiki/Lab_color_space (1.3.2009).