Mobilna aplikacija za praćenje vodostaja rijeke obradom slike s bespilotne letjelice

Semialjac, Kruno

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:055558

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplomski studij računarstva

MOBILNA APLIKACIJA ZA PRAĆENJE VODOSTAJA RIJEKE OBRADOM SLIKE S BESPILOTNE LETJELICE

Završni rad

Kruno Semialjac

Osijek, 2018.

SADRŽAJ

1. UVOD 1
   1.1. Zadatak završnog rada 1
2. PRIMJENA BESPILOTNIH LETJELICA U NADZORU RAZLIČITIH POVRŠINA 2
   2.1. Vodostaj rijeka 5
      2.1.1. Poplave 6
      2.1.2. Zaštita od poplava 6
   2.2. Postojeće aplikacije za mjerenje vodostaja 7
   2.3. Ideja aplikacije za mjerenje vodostaja 8
3. MODEL MOBILNE APLIKACIJE 9
   3.1. Prikaz rada aplikacije 9
   3.2. Model snimanja slike bespilotnom letjelicom 10
   3.3. Model obrade slike 13
      3.3.1. Pronalaženje mjere vodostaja 16
4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE 20
   4.1. Korišteni alati, tehnologije i programski jezici 20
      4.1.1. Android Studio 20
      4.1.2. Dron Phantom 3 Advanced 21
      4.1.3. DJI Development KIT 21
      4.1.4. DJI GO aplikacija 22
      4.1.5. OpenCV Library 22
   4.2. Ključni dijelovi koda 22
      4.2.1. Implementacija knjižnice openCV u Android Studio 22
      4.2.2. Prikaz izgleda sučelja aplikacije 25
      4.2.3. Prikaz osnovnih funkcija aplikacije 26
      4.2.4. Prikaz korištenih paketa u aplikaciji 31
   4.3. Programska arhitektura 32
   4.4. Prikupljanje i spremanje podataka 32
5. NAČIN RADA I KORIŠTENJE APLIKACIJE 34
   5.1. Korištenje aplikacije 34
   5.2. Prikaz primjera obrade slike u aplikaciji 37
   5.3. Analiza rada sustava 38
6. ZAKLJUČAK 40
LITERATURA 41

SAŽETAK 43
ABSTRACT 44
ŽIVOTOPIS 45
PRILOZI 46

1. UVOD

Tema ovog završnog rada je mjerenje i praćenje vodostaja rijeke bespilotnom letjelicom putem mobilne aplikacije koja obradom slike i određenim postupcima daje visinu vodostaja. Ljudi mjere vodostaj voda - mora, jezera ili rijeka unazad više od 200 godina. Zanimanje za mjerenje vodostaja pokazalo se potrebnim kako bi se mogla predvidjeti opasnost od mogućih poplava, a samim time i djelovati da se zaštite ljudski životi te imovina. Razvojem tehnologije dolazi do značajnih promjena u svijetu koje snažno utječu na svakodnevni život ljudi. Nova tehnološka postignuća prvenstveno mijenjaju društvena gledišta naših života, olakšavaju svakodnevne poslove i omogućavaju pregršt informacija koje služe te pomažu u daljnjem razvoju. Razvojem bespilotnih letjelica može se pristupiti različitim lokacijama, pa tako i onima kojima ljudi jednostavno (bez upotrebe bespilotne letjelice) ne mogu pristupiti. Također, mogu se snimati slike, audio i videozapisi iz različitih perspektiva što omogućuje bolji pregled promatrane površine. Cilj ovog završnog rada je prikazati način mjerenja vodostaja upotrebom tehnologije kao što su bespilotne letjelice. Korištenjem kamere letjelice snima se fotografija koja se potom obradi pomoću programskog koda napisanog u integriranom razvojnom okruženju.

U drugom poglavlju opisana je primjena bespilotnih letjelica u nadzoru površina kao što su ravnice, planine i vodene površine te mjerenje vodostaja rijeka u svrhu dojavljivanja opasnosti od poplava. Također, bit će objašnjeni najčešći razlozi nastanka poplava te preventivne mjere zaštite od poplava. Nadalje, bit će prikazane postojeće aplikacije za mjerenje vodostaja kao i ideja ovog završnog rada. U trećem poglavlju opisan je model Android aplikacije, gdje će se prikazati rad sustava te način snimanja i obrade slike s bespilotne letjelice. Četvrto poglavlje sastoji se od programskog rješenja mobilne aplikacije. Prikazani su korišteni alati, tehnologije i programski jezici, izgled programske arhitekture, način pohrane podataka u bazu podataka, te ključni dijelovi programskog koda. U petom poglavlju prikazano je korištenje aplikacije, rezultati obrade i analiza rada sustava.

1.1. Zadatak završnog rada

U radu treba opisati mogućnosti raspoložive bespilotne letjelice, način prijenosa slike na mobilni uređaj te mogućnosti njene daljnje obrade. Nadalje, treba osmisliti i programski

implementirati postupak obrade slike koji će u mobilnoj aplikaciji omogućiti praćenje vodostaja rijeke te alarmiranje korisnika. Sustav treba ispitati i analizirati.

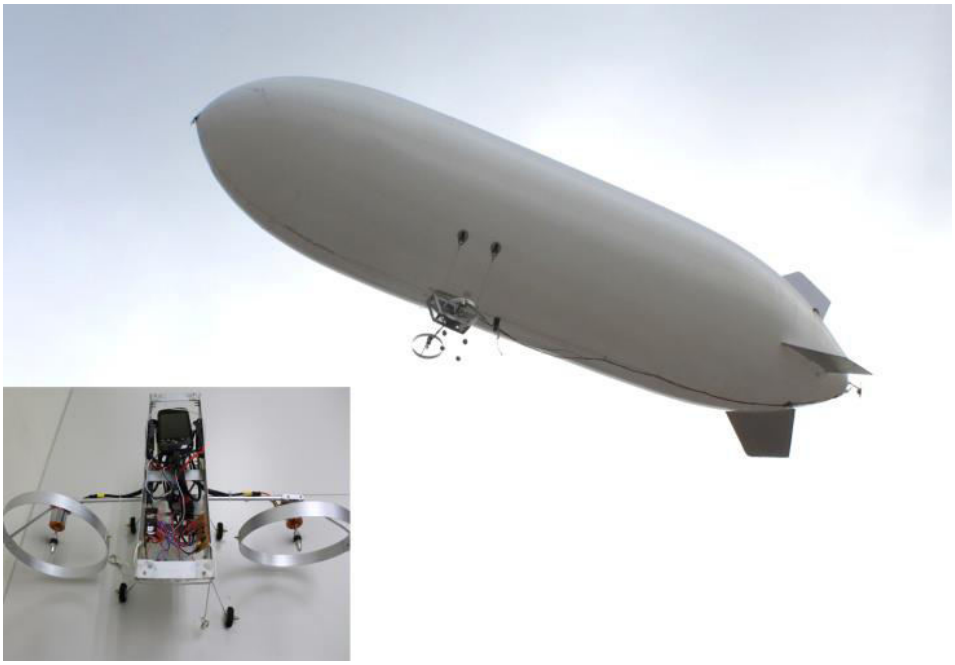
2. PRIMJENA BESPILOTNIH LETJELICA U NADZORU RAZLIČITIH POVRŠINA

Prve bespilotne letjelice počele su se razvijati u vojne svrhe, kako bi se mogao promatrati neprijateljski teritorij s minimalnim ljudskim gubitcima. Prve zabilježene bespilotne letjelice bili su austrijski leteći baloni davne 1849. godine koji su prenosili eksploziv prema talijanskom gradu Veneciji. Tada su ti baloni na veoma domišljat način imali svrhu razaranja neprijatelja. Iako se baloni danas ne smatraju bespilotnim letjelicama, to je bila tehnologija koju su Austrijanci mjesecima razvijali, što je dovelo do daljnjeg razvoja letjelica. Prije početka Prvog svjetskog rata, nastankom prvog aviona na vlastiti pogon, dolazi do velike ekspanzije za daljnjim razvojem takvih strojeva i njezinom tadašnjom tehnologijom. Dolaskom rata, avion je izgubio sve druge moguće svrhe osim - da postane smrtonosno oružje. Tako ih je 1915. godine Velika Britanija prva koristila za mapiranje teritorija koristeći se fotografijama snimljenih iz zraka. U ono vrijeme to je bio veliki uspjeh jer su uspjeli uhvatiti 1500 različitih mapa i to teritorija koja je držala njemačka vojska pod kontrolom. Sjedinjene Američke Države[1] i Francuska bile su prve zemlje koje su počele razvijati avione bez pilota. Takve letjelice su bile kontrolirane radio vezom a imale su osobine kao „zračni“ torpedo. Nakon rata Američka ratna mornarica nastavila je eksperimentirati s razvojem bespilotnih letjelica te tako napravila prvi ikada dron koji je bio veličine aviona dugačkog par metara. Kasnije su uspjeli usavršiti i avion bez pilota koji je bio upravljan daljinskim upravljačem. Bespilotne letjelice prethodno su bile poznate kao nepredvidljive i veoma skupe igračke no to se sve mijenja 1980. godine kada je SAD stvorio program poznat kao „Pioneer UAV“ koji je ispunjavao potrebu za razvojem jeftinijih letjelica za izviđanje. Od tada dolazi do razvoja brojnih modela koji se razlikuju po veličini, brzini i namjeni. 1990. godine stvoreni su takozvani minijaturni ili mikro dronovi koji su veličine ping pong loptice čija je svrha, upravo zbog svoje male veličine, bila špijuniranje neprijatelja. Nekoliko godina poslije toga razvijena je poznata bespilotna letjelica „Predator“ koja je služila u Američkim vojnim misijama za vrijeme rata na Balkanu, Afganistanu i Siriji. [1]

[1] U kasnijem tekstu - SAD

Nakon razvijanja bespilotnih letjelica u vojnoj industriji i njezino korištenje istih, mnoge druge grane industrija su počele koristiti takozvane dronove. U poljoprivredi se upotreba dronova pokazala iznimno korisnom. Naime oni svaki dan lete preko mnogih obradivih površina i pritom snimaju slike s posebnim kamerama u visokoj rezoluciji poput (FPI) spektralnih kamera temeljena na interferometru Fabry-Perot[12] koja omogućuje prikupljanje spektrometrijske blokove slika, a koje se kasnije analiziraju metodama opisanih u referenci [12] putem računalnog sustava. Te metode su dale realne i zadovoljavajuće rezultate usprkos vremenskim neprilikama. Dobiveni rezultati govore o uspješnosti usjeva, stanju tla polja i procjeni biomase te o mogućim rješenjima ako usjev nije dobar.

Dronovi se koriste i u nadzoru brojnih zastarjelih kritičnih infrastruktura, kao što su plinovodi i naftni cjevovodi koji većinom prolaze kroz osjetljivi dio okoliša. Svako oštećenje infrastrukture može znatno štetiti fauni i flori okoline te doći do otrovanja zaliha vode. Takav tip nadzora omogućeno je sustavom i razvijenim algoritmom: dodjeljivanja stanica za punjene i njihova preraspodjela po području, prijava drona na stanice za punjenje i izračun duljine njihovih putanja [11]. Naime takve bespilotne letjelice imaju vremenski kratak let (oko 20min) zbog njihovih „slabih“ baterija, dok se infrastrukture plinovoda znaju protezati po desetak kilometara. Drugo pak rješenje za takav tip nadzora je BlimpProbe (Sl. 2.1) preuzeta iz [10]. Vrsta bespilotnog cepelina koji je jeftiniji od dronova, može lebdjeti iznad određenog područja u dužim vremenskim intervalima pri maloj potrošnji energije, te ima prednost što može nositi puno dodatne opreme poput senzora, baterija i posebnih kamera.



Sl. 2.1. Prikaz BlimpProbe bespilotne letjelice [10].

Osim u industriji, bespilotne letjelice koriste se i u slučaju elementarnih nepogoda i katastrofa. Prilično savršen nadzor terena omogućuju ugrađene infracrvene kamere i *Night Vision* kamere. Kako su dronovi postali široko dostupni u konvencionalnim trgovinama i to po prihvatljivim cijenama, upotrebom više letjelica mogu se obraditi velike površine u relativno malom vremenskom roku što pridonosi bržem reagiranju i procesu spašavanja. Takav tip planskog obrađivanja površina je osmislio tim stručnjaka iz Japana [7]. Oni predlažu da kada dođe do katastrofe, svaki dostupni dron u obližnjem kućanstvu spoji se na sustav, koji ih kasnije raspodjeli kako bi što efikasnije snimili područje pogođeno nepogodom. Kako bi se to postiglo, letjelice bi se morale povezati putem mobilnih ad hoc mreža (tip bežičnih mreža koje povezuju mobilne uređaje, najčešće se koriste u vojne svrhe).

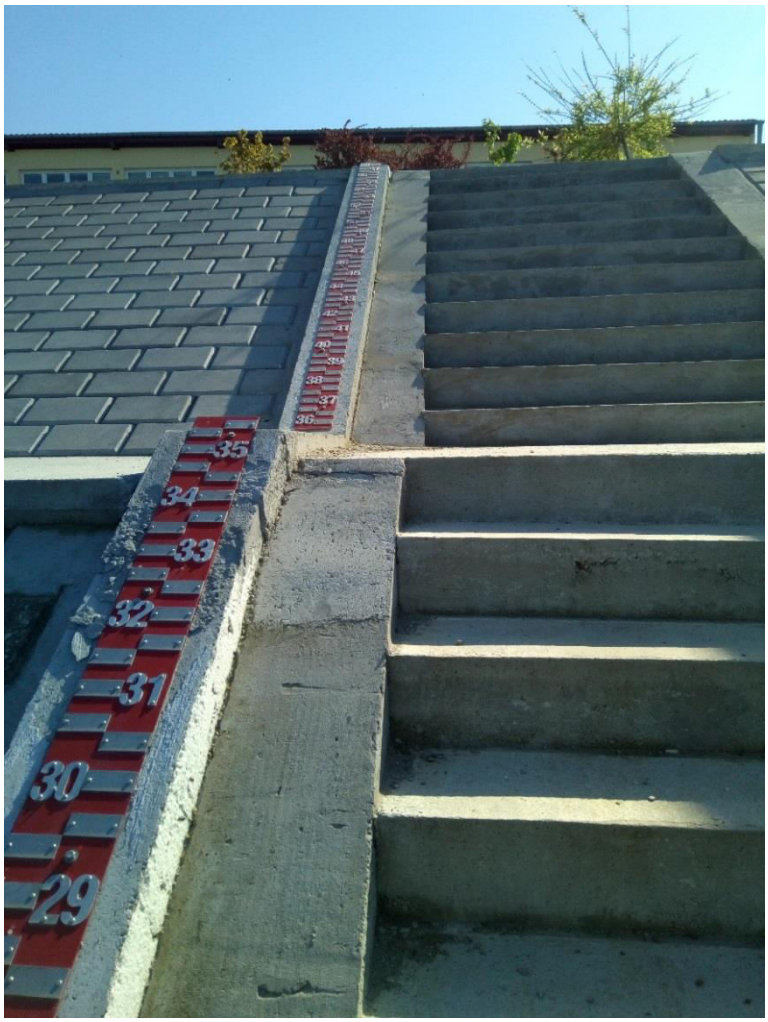
Bespilotne letjelice se mogu koristiti i u službama za spašavanje. Zbog njihovih malih dimenzija, dronovi mogu pristupiti na teškom dostupnim mjestima poput planinskih klanaca, kanjona i mnogih drugih. Također već spomenutim specijalnim kamerama može se analizirati mjesto unesrećene osobe i predvidjeti moguće nadolazeće prepreke i opasnosti. Pritom se troši jako malo resursa prije negoli se krene u akciju spašavanja.

Osim u nadziranju površina, infrastruktura, djelovanja u slučaju nepogoda i spašavanju ljudskog roda, bespilotne letjelice mogu poslužiti i u mnogim geografskim granama, poput

kartografije i hidrografije. Iako postoje brojne satelitske snimke zemljine površine, s njima je teško raditi analize zbog raznih problema. Jedan od tih problema je što se satelit nalazi na velikoj udaljenosti od zemlje, pa samim time slika nema „dubinu“, pa slike izbočina (poput planina i brda) ili nekih udubina izgledaju kao da su spljoštene, odnosno ne može se razaznati niti predvidjeti stvarna veličina gledanog objekta. Drugi problem je kada, prilikom približavanja ili uvećanja slike na dosta malu udaljenost od zemljine površine, slika nije dovoljno izoštrena da bi putem obrade i analize dala konkretna rješenja. Postoje i mnogi drugi problemi poput cijene satelita i njegovog lansiranja u svemir radi mapiranja površine, neke slike pojedinih lokacija su stare i preko par godina (nemogućnost praćenja objekta u stvarnom vremenu) i mnogih drugih. Kao rješenje tih problema su upravo bespilotne letjelice, koje mogu zapravo nadopuniti nedostatke satelitskih slika i stvoriti kompletnu sliku na potpuno novu razinu. Takvim tipom mapiranja, uz pomoć drona, uspjeli su izvesti talijanski stručnjaci koji su analizirali rijeku i njezin estuarij, ponašanje iste u slučaju obilnih kiša i prikaz okolne površine koja bi mogla biti poplavljena [8].

2.1. Vodostaj rijeka

Razina vode ili vodostaj je visina površine vode od određene točke koja se zove kota, a koja je zapravo geodetskim izračunima određena kao nadmorska visina područja u kojem se mjeri vodostaj. Prema tome, vrijednosti vodostaja mogu biti i negativne jer se dno korita rijeke može nalaziti ispod predodređene kote, a da pritom dubina rijeke može biti i po par metara. Osim za mjerenje visine površine vode poput rijeka, jezera i mora, vodostaj služi kao informacija prikaza visine vode u određenim spremnicima, izračun dubina korita i izrada navigacijskih mapa za plovidbu brodova i mnogih drugih osobina. Postoji nekoliko uobičajenih metoda za mjerenje vodostaja od kojih svaka ima svoje prednosti i nedostatke mjerenja, troškova, rada i održavanja. U Republici Hrvatskoj koristi se vodomjerna letva (Sl. 2.2) kao naprava za mjerenje razine vode. Vodostaj se mjeri svakih sat vremena, te se uz to prati trend kretanja razine vode i brzina protoka rijeke kako bi se predvidjelo njezino kretanje.



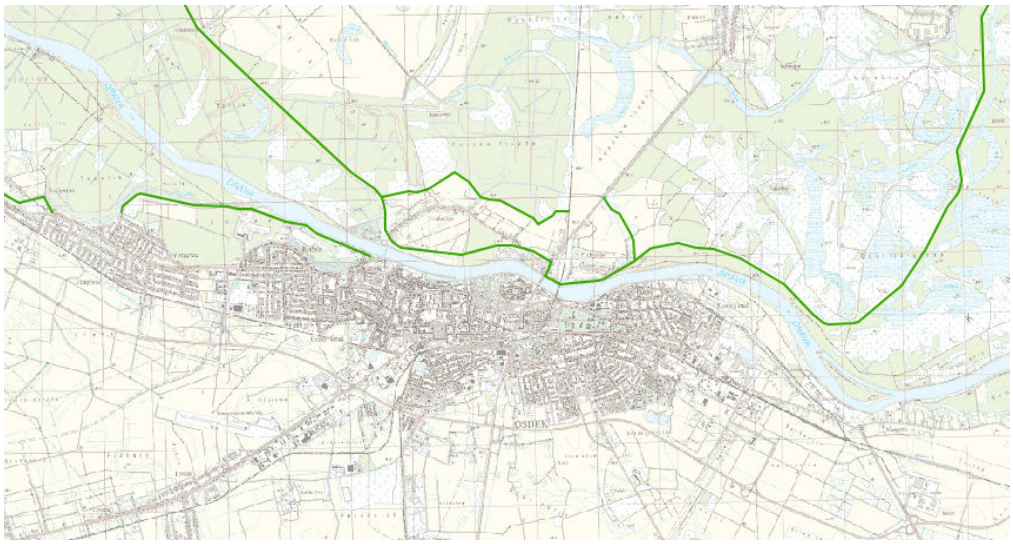
Sl. 2.2. Prikaz vodomjerne letve na rijeci Dravi.

2.1.1. Poplave

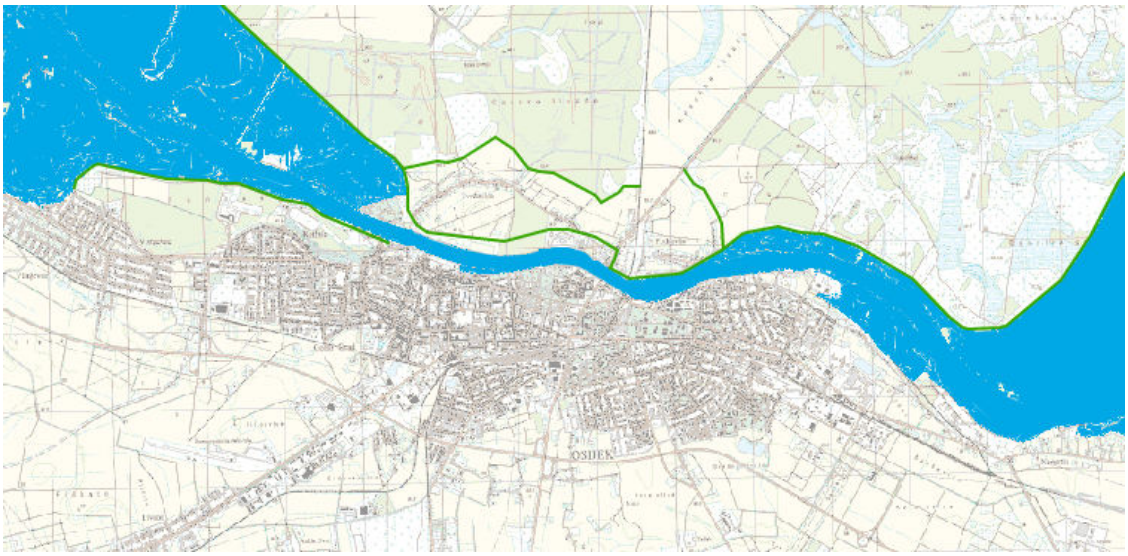
Jedan od razloga zašto se mjeri i redovito prati tijekom dana vodostaj rijeka je kako bi se pravovremeno upozorilo na mogućnost dolaska velikih količina voda. Kada se ta ista količina vode izlije iz korita i dođe na područje na kojem ne prevladava voda (npr. polja, livade, šume, naselja, itd.), ta se pojava zove poplava [6]. Postoje mnogi načini kako dolazi do poplava. Najčešći razlozi su ogromne količine padalina, otapanje snijega i leda nastalih u zimskim mjesecima u planinskim i brdskim područjima, stvaranje brana s lošom kontrolom protoka vode i rezultat djelovanja potresa na moru koji stvaraju valove visoke i po par metara. Poplave uzrokuju ozbiljne štete kućama, cestama, poslovnim objektima, a ono najgore je što oduzimaju ljudske živote.

2.1.2. Zaštita od poplava

Svaka država ima zavod koji prati kontrolu razine vode u rijekama i razrađen plan u slučaju opasnosti od poplava. Glavne zaštite u naseljenim mjestima poput gradova su brane i nasipi duž cijele obale. Brane su osmišljene i projektirane tako da sprečavaju i kontroliraju izljev vode na određenim mjestima, dok nasipi kontroliraju jednu stranu obale (onu gdje je naseljena ljudska populacija). Kao najvažnije sredstvo za obranu od poplava služe karte (Sl. 2.3 i Sl. 2.4) koje predočuju rizik i opasnost od poplava [13]. Pomoću diskretnih modela prethodnih poplavnih događaja i modelom koji su se ti događaji vremenski odvijali generiraju se statičke i dinamičke poplavne karte kroz simulaciju [14]. Također, na karti je moguć uvid gdje se nalaze nasipi i koja su područja najugroženija ako dođe do poplava.



Sl. 2.3. Prikaz lokacija nasipa u okolici grada Osijeka [13].



Sl. 2.4. Prikaz karte opasnosti od poplava s velikom vjerojatnošću od poplavljivanja [13].

2.2. Postojeće aplikacije za mjerenje vodostaja

Trenutni život ljudi odvija se u razdoblju gdje su pametni telefoni postali dio svakodnevnog života. Rijetko koja osoba nema uz sebe *smartphone* jer su upravo oni postali osnovni uređaji za komuniciranje diljem svijeta. Dobitkom takve vrste tehnologije javlja se potreba za boljom programskom podlogom, pa se tako izvlače svi maksimumi kako bi se pružile najbolje usluge korisnicima. U svijetu aplikacija, na kreativan i inovativan način, programeri nude usluge od zabave (poput igrica, prikazivanje filmova, puštanje pjesama), društvenih mreža (prijenosa poruka, slika, itd.) do onih koji nude medicinske, kuharske priručnike i informacije poput vijesti i vremena. Prema nekakvom pravilu u današnje vrijeme moguće je pronaći doslovno sve što jednoj osobi zatreba i to samo u par klikova. U RH postoje brojne stanice za mjerenje vodostaja. Svoja mjerenja i bilješke stavljaju na web stranice, te na taj način omogućuju internet korisnicima iste. Programeri putem aplikacijskog programskog sučelja (API) mogu te informacije koristiti u svojim aplikacijama. Jedna od brojnih aplikacija koju pruža takvu vrstu informacija je aplikacija „Vodostaji“ od Hrvatskih Voda koja je i službeni vlasnik mjernih stanica u Hrvatskoj [2]. S pozitivnim recenzijama aplikacija je ocijenjena vrlo dobrom pružajući time u bilo kojem trenutku vodostaj svih rijeka, mogućnost poplave koja je kategorizirana u 5 kategorija s pripadajućim mjerama, brzinom protoka i trendovima kretanja razine vode. Postoje i druga kreativna rješenja poput projekta dojave poplava putem SMS i MMS kojeg je razvio tim stručnjaka s malezijskog sveučilišta nazvan „FLOWS“ [6]. Glavni cilj tog sustava je praćenje stanja poplava i mjerenje razine vode na područjima visokog

rizika. Sustav bi bio sposoban upozoriti stanovnike i lokalne vlasti o trenutnom stanju poplava kako bi se mogle poduzeti pravovremene akcije spašavanja. Cilj je postignut tako što sustav mjeri i sprema vrijednost razine vode te je kategorizira u tri različita stupnja. *Hardware* i *software* su uspješno integrirani da vrlo dobro međusobno komuniciraju i da daju očekivane rezultate poput slika, grafikona s trenutnim podacima i mogućnost najbližeg skloništa [6].

2.3. Ideja aplikacije za mjerenje vodostaja

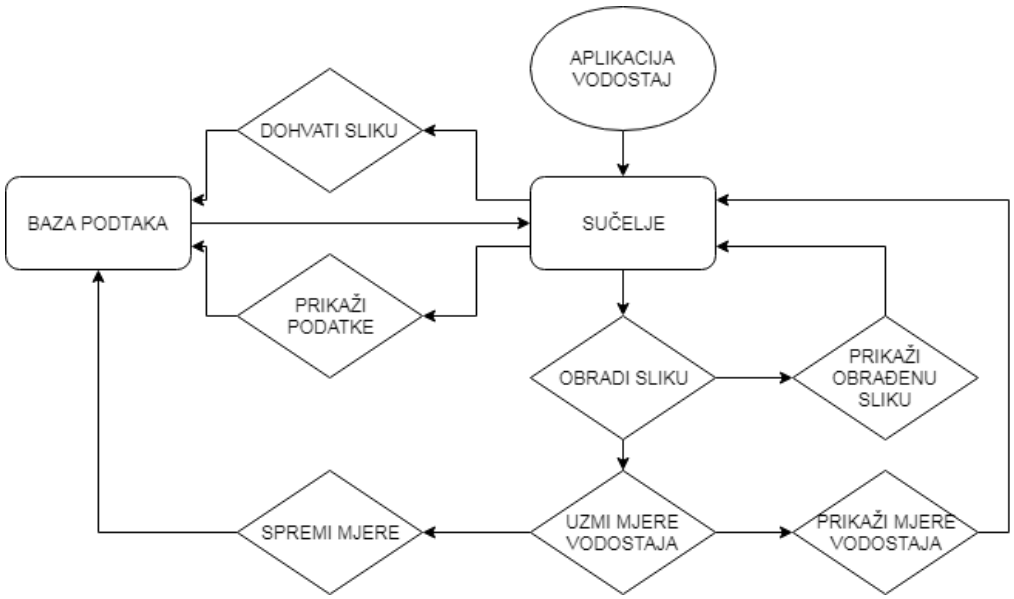
Na pitanje: „Što kada nestane internet?“ što u današnje vrijeme nije nemoguće, pogotovo u slučaju elementarnih nepogoda, dok poplave mogu biti veoma razdorne u prilogu primjer rijeke Save u okolici Gunje. Dolazi se do ideje kako iskoristiti i ukomponirati pametni telefon kojeg većina ljudi ima i tehnologiju kao što su bespilotne letjelice poput dronova koja se trenutno probija na tržištu. Korištenjem drona može se obuhvatiti velika količina površine koja je potrebna da se snimi cijela širina rijeke, dok putem aplikacije koja može pružiti pregršt usluga, mogu se iskoristiti dobivene informacije s bespilotne letjelice. Način na koji se može doći do vodostaja kao i cjelokupan proces obrade i analiza slike prikazano je pomoću modela koji se nalazi u 3. poglavlju.

3. MODEL MOBILNE APLIKACIJE

Važno je prikazati rad aplikacije kroz modele kako bi se što bolje shvatilo funkcioniranje određenih dijelova sustava. Mjerenje vodostaja rijeke pomoću bespilotne letjelice opisuje se kroz model snimanja slike i model obrade slike. U modelu snimanja slike opisan je cjelokupan rad s Phantom 3 Advanced dronom od dolaska na mjesto snimanja do toga što treba sve obuhvatiti dok snima, pritom treba pripaziti i na karakteristike leta i upravljanjem istim. Model obrade slike je poprilično složen način koji se odvija u dva koraka. Prvi korak je prikaza rada funkcija iz OpenCV knjižnice koje obrađuju sliku, dok je drugi korak prikaz na koji način se dolazi do algoritma koji preračunava rezultat obrade slike u računicu koja opisuje vodostaj rijeke.

3.1. Prikaz rada aplikacije

Prikaz rada sustava aplikacije za mjerenje vodostaja dijagramom toka s osnovnim funkcijama dan je na slici 3.1.



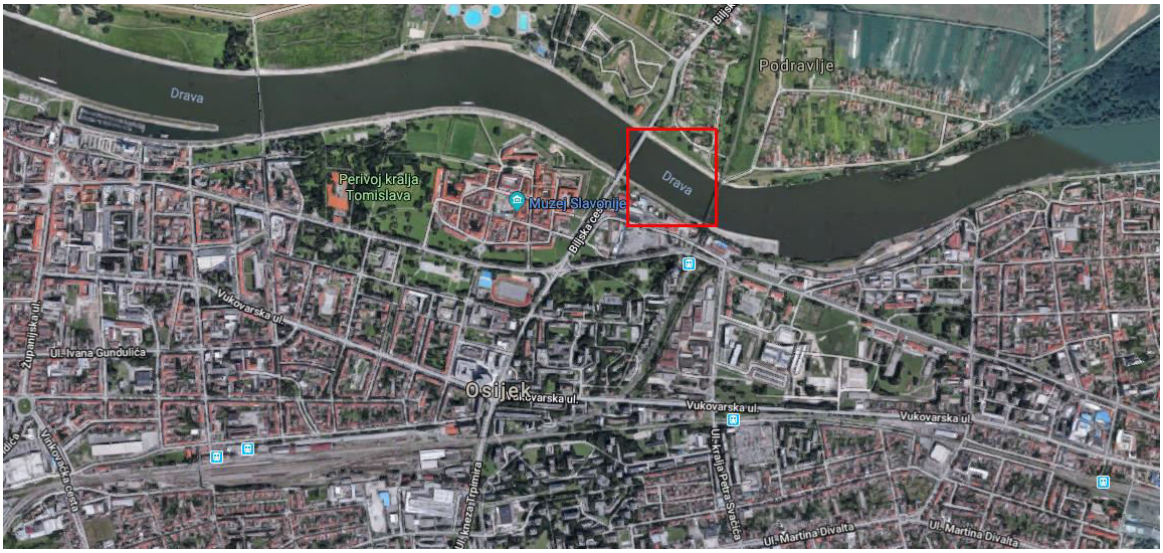
Sl. 3.1. Dijagram toka rada aplikacije.

Aplikacija se sastoji od sučelja na kojem su prikazane funkcije: prikaži podatke i dohvati sliku. Odabirom tih funkcija, aplikacija otvara bazu podataka i u njoj dohvaća prethodno spremljene podatke (datum snimanja vodostaja, vodostaj, itd.) ili otvara galeriju slika kako bi odabrali sljedeću za obradu. Treća funkcija sa sučelja je obradi sliku koju procesor mobilnog uređaja izvodi čitanjem naredbi napisanog koda. Zatim otvara novi prozor na kojem prikazuje obrađenu sliku s novim funkcijama: uzmi mjere vodostaja i spremi mjere.

Uzimanjem mjera, obrađena slika prolazi kroz algoritam. Nakon što algoritam završi s analiziranjem, na ekran se ispisuje vrijednost vodostaja. Odabirom funkcije spremi mjere, vrijednosti se spremaju u bazu podataka, zatvara se prozor i prikazuje se ponovno početno sučelje.

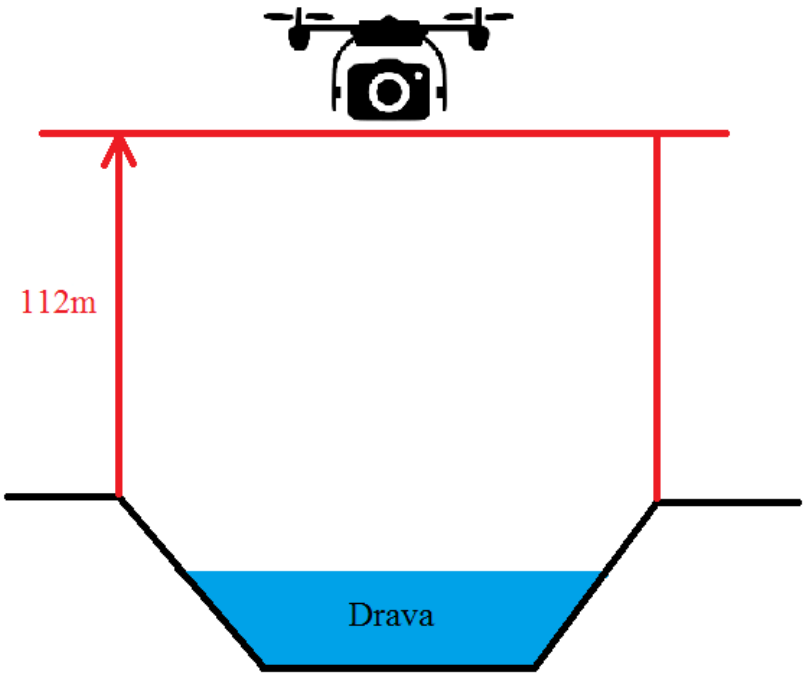
3.2. Model snimanja slike bespilotnom letjelicom

Snimanje slike bespilotnom letjelicom se izvršava na rijeci Dravi u Osijeku. Snimanje se izvršava na određenom dijelu Drave između cestovnog i željezničkog mosta prema slici 3.2 na kojem su poznate karakteristike rijeke i obale. Na tom području lijeva i desna obala rijeke su međusobno usporedne što je važno prilikom obrađivanja slike metodom koju opisuje ovaj završni rad. Širina korita rijeke je najmanja na tom dijelu što također ide u korist prilikom snimanja, jer bi dron u suprotnom morao ići na veće visine u odnosu na vlastite mogućnosti.



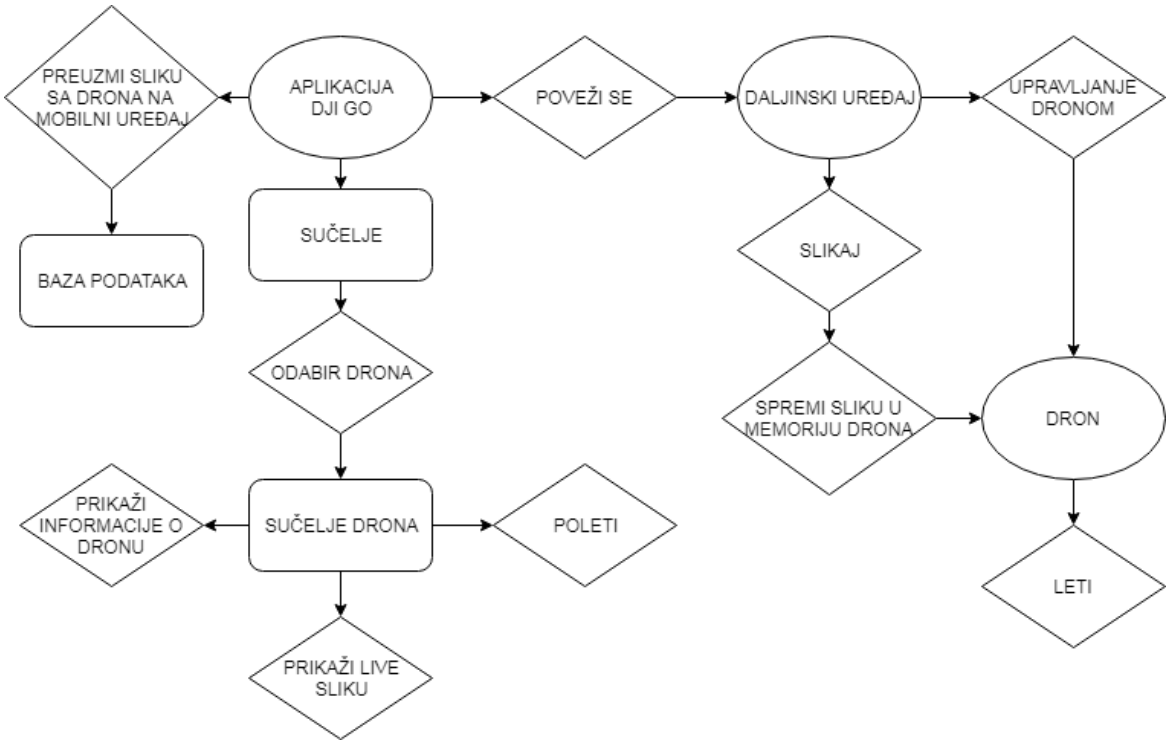
Sl. 3.2. Markirano područje u kojem se mjeri vodostaj rijeke Drave.

Kako bi mjerenje vodostaja bilo točno, dron prilikom snimanja mora biti na određenoj visini koja iznosi 112m (na toj visini dron uspije obuhvatiti rijeku zajedno sa obalom u jednom kadru, detaljnije objašnjeno u poglavlju 3.3.1) od površine zemlje i mora biti u sredini, između obalnih strana kao što je prikazano na slici 3.3.



Sl. 3.3. Prikaz položaja drona prilikom snimanja slike.

Kako bi se let omogućio s Phantom 3 Advanced dronom potrebno je uz njega imati upravljački uređaj (preko njega se vrše osnovni manevri leta, kontrolira se gibanje kamere i snima fotografija/video) i mobilni uređaj s instaliranom aplikacijom DJI GO [15]. Prikaz rada sustava aplikacije DJI GO za upravljanje dronom prikazano je dijagramom toka na slici 3.4. Paljenjem aplikacije, mobilni uređaj povezuje se preko USB kabla na upravljački uređaj (na slici 3.4 daljinski uređaj). Otvara se sučelje preko kojeg se odabire određena bespilotna letjelica iz DJI kataloga. Pritiskom na kameru odabranog drona otvara se novo sučelje u kojem se prikazuje video prijenos uživo, karta s lokacijom gdje se nalazi dron i gdje se nalazi korisnik, brojne informacije o dronu i opcija preko koje dron kreće sa poletanjem. Na upravljačkom uređaju nalaze se tipke za slikanje i snimanje. Pritiskom na te tipke dron sprema podatke na SD karticu koja se nalazi na dronu. Nakon leta u aplikaciji DJI GO se odabere opcija „Editor“ u kojem se mogu pronaći slike uslikane dronom te ih se preuzme u punoj rezoluciji i spremi u bazu podataka koja se nalazi na mobilnom uređaju (kasnije se slike automatski preuzimaju prilikom završetka leta).



Sl. 3.4. Dijagram toka rada aplikacije za upravljanje dronom.

Pomoću kamere na bespilotnoj letjelici snimi se slika kao što je prikazano na slici 3.5 na način da se uhvati cijela širine rijeke. Na dobivenoj slici može se vidjeti rijeka, projektirana kamena obala i zelena okolina.



Sl. 3.5. Prikaz primjera snimke rijeke Drave s bespilotne letjelice (ispravno uslikano).

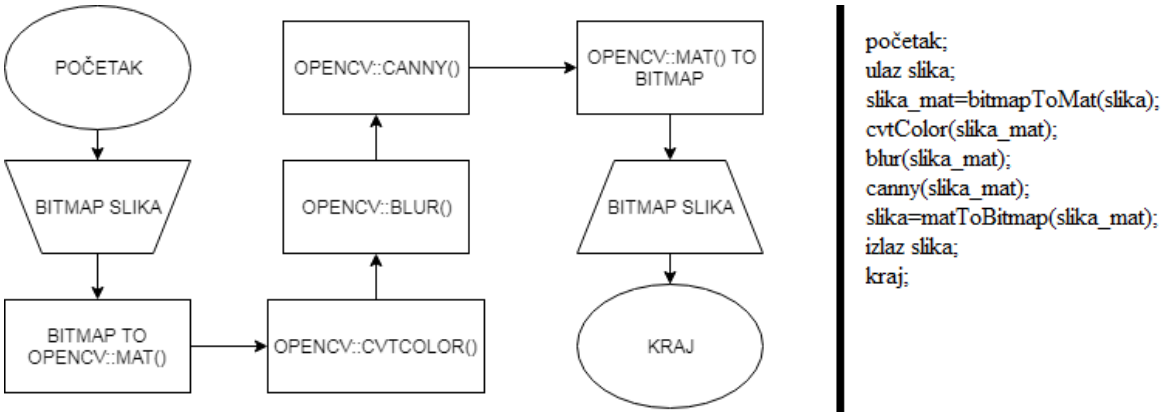
Prilikom snimanja slike važno je da rijeka bude okomita u odnosu na dno slike kako bi obrada bila uspješnija. Ako se snimi fotografija kao na slici 3.6. doći će do prevelikog odstupanja vrijednosti vodostaja od stvarne vrijednosti razine vode. Do odstupanja dolazi zato što vodostaj linearno ovisi o širini rijeke, dok će aplikacija širinu rijeke krivo učitati ukoliko je ona zakrivljenija u odnosu na dno slike.



Sl. 3.6. Prikaz primjera snimke Drave s bespilotne letjelice (neispravno uslikano).

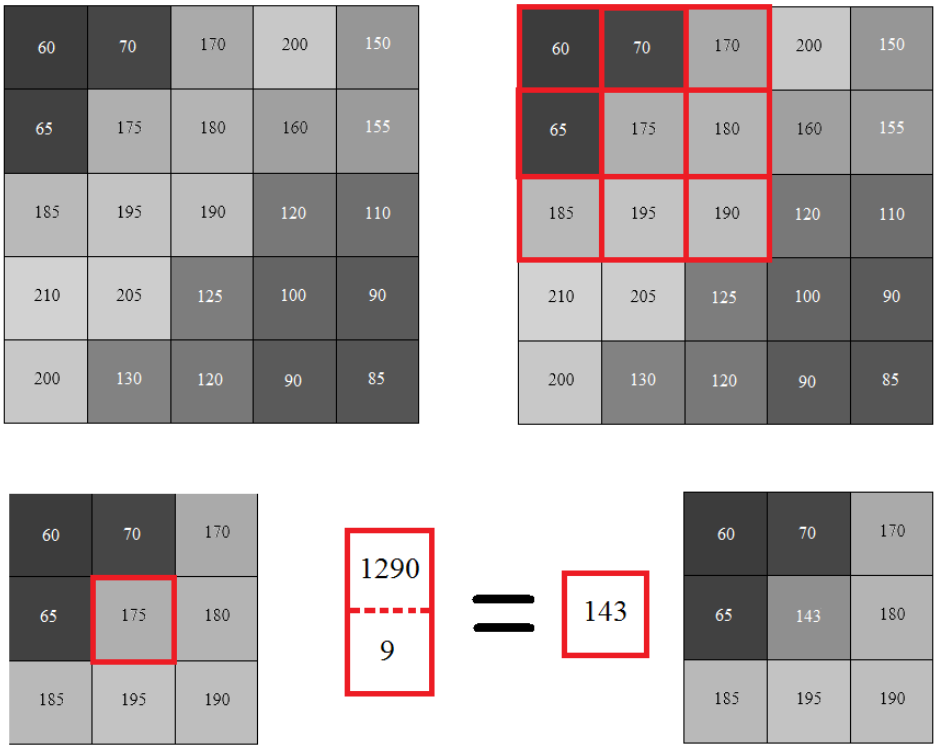
3.3. Model obrade slike

Na slici 3.7 prikazan je dijagram tijeka obrade slike napisan pomoću pseudokoda. Aplikacija učitava fotografiju iz memorije mobilnog uređaja u obliku bitmap slike. Takvim tipom formata slike dobije se na uvid: od kojih se piksela sastoji slika, na kojoj poziciji se nalazi pojedini piksel i koje su njegove RGB vrijednosti (aditivni model boja koji govori koliko piksel sadrži crvene, zelene i plave boje). Nadalje, slika se formatira u *Mat* tip podataka kojeg razumije OpenCV knjižnica kako bi se mogle izvršavati njezine funkcije za obradu slike. Prvo se izvodi funkcija *cvtColor* koja pretvara izabranu fotografiju u crno-bijelu sliku, gdje su svjetlije boje izraženije bijelom, dok su tamnije izraženije crnom bojom. S tom funkcijom svaki piksel na fotografiji može se promatrati kao da ima jednu vrijednost (zato što su vrijednosti crvene, zelene i plave u tom slučaju jednake), 0 označava crni piksel, dok 255 označava potpuno bijeli piksel.



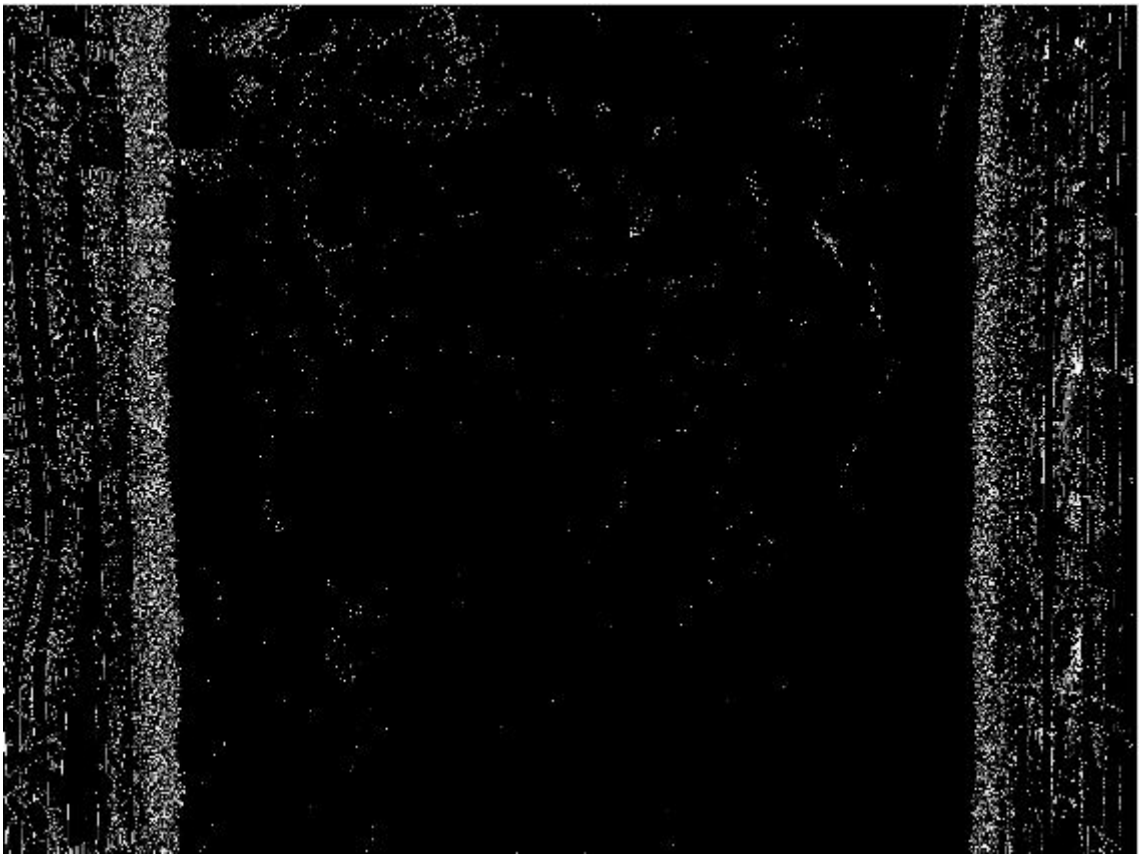
Sl. 3.7. Prikaz pseudokoda i dijagrama toka obrade slike.

Sljedeća funkcija koja se izvodi zove se *blur* koja ima svojstvo zamagljivanja slike, a koja se koristi kako bi se filtrirale sitne smetnje. Ta funkcija zahtjeva 2 parametra koja označavaju broj redaka i stupaca promatrane rešetke nazvanom jezgra. Slika 3.8 prikazuje skupinu piksela s fotografije za koju je odabrana jezgra 3x3. Za primjer (Sl. 3.8) obrađuje se piksel s vrijednosti 175 koji se nalazi u središtu jezgre, na način da se zbroje svi okolni pikseli i podjeli s 9 (3 redaka * 3 stupaca), odnosno uzima se srednja vrijednost, te mu se dodjeljuje dobivena vrijednost tog količnika. Funkcija se izvodi za svaki piksel sa slike.



Sl. 3.8. Obrada piksela s *blur* funkcijom.

Posljednja funkcija se zove *canny* koja implementira detektor rubova na fotografiji. Funkcija u sebi sadrži algoritam koji zahtjeva tri kriterija: nisku stopu pogrešaka (odnosno detekcija samo postojećih rubova), dobru lokalizaciju (minimizacija udaljenosti između otkrivenih rubnih piksela i postojećih rubnih piksela) i minimalan odziv (odziv jednog detektora po otkrivenom rubu). Prvo se pronalazi gradijent intenziteta slike pomoću procedure analognoj Sobel operatoru [16]. Zatim se primjenjuje ne-maksimalna supresija koja uklanja piksele koji se ne smatraju dijelom ruba. Ostaju samo tanke linije piksela koji se svrstavaju kao kandidati za rubne linije. U posljednjem koraku *canny* funkcije, koriste se dva praga: gornji i donji prag. Ako je gradijent piksela veći on gornjeg praga, piksel se prihvaća kao rubni. Ako je gradijent piksela između gornjeg i donjeg praga, piksel se prihvaća samo ako je isti piksel povezan s pikselom iz gornjeg praga. Svi pikseli koji su manji o donjeg praga se odbacuju. Kako bi bilo ispravno detektirano, omjer gornjeg i donjeg praga mora biti 2:1 ili 3:1. Obradom slike (Sl. 3.5.) opisanim funkcijama i procedurama kao rezultat dobiju se istaknuti rubni pikseli na crnoj pozadini prikazani na slici 3.9.



Sl. 3.9. Prikaz slike obradom funkcija *cvtColor*, *blur* i *canny*.

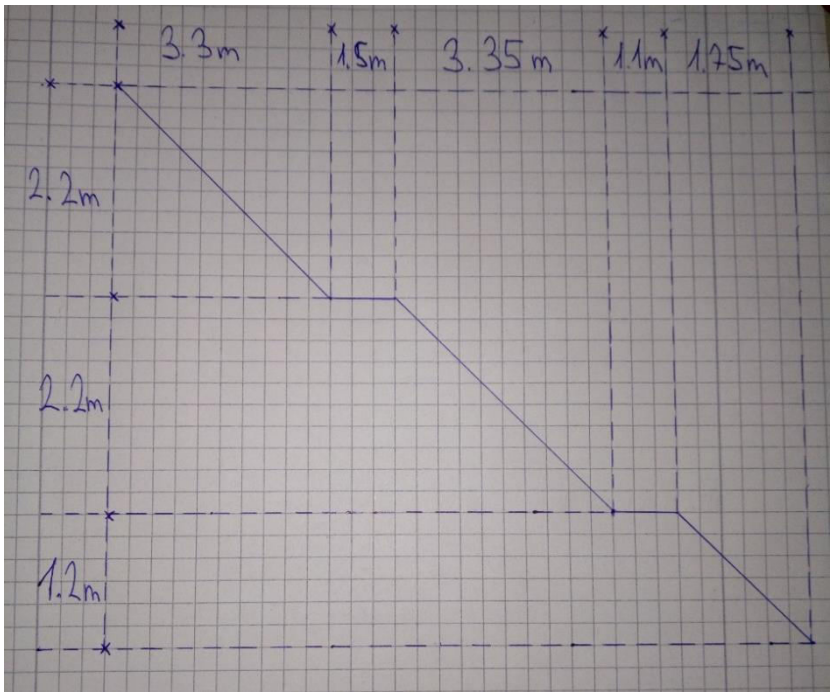
Može se uočiti da iako se koriste funkcije kako bi se smanjile smetnje, one se nisu u potpunosti uklonile, niti se mogu u potpunosti ukloniti zbog više razloga. Prvi razlog su bijeli pikseli koji se nalaze na području rijeke. Oni predstavljaju odsjaj sunca na površini vode. Nemoguće je uslikati Dravu tijekom dana a da pritom nema sunčevog odsjaja (op. ako je oblačno zrcalit će se oblaci). Drugi razlog su strana tijela u vodi poput komada drveća, brodova i mnogih drugih. Sve to funkcija *canny* prepoznaje kao objekte te im ocrtava rubove.

3.3.1. Pronalaženje mjere vodostaja

Sljedeći korak aplikacije je prepoznavanje obala rijeke. Dobivena slika formatira se nazad iz *Mat* tipa u bitmap oblik. Algoritam radi na način da se promatra stupac po stupac piksela sa zadane slike. Kako je već rečeno, svaki piksel ima svoju x i y koordinatu. Počevši od stupca iz sredine slike prema desno, analizira se koliko bijelih piksela sadrži isti. Kada algoritam otkrije veću količinu bijelih piksela, više od 1/4 ukupnih piksela u tom stupcu (kako bi odbacio nakupine piksela uzrokovanje smetnjama), on taj stupac prepoznaje kao rub obale, te sprema njegov x položaj. Zato je važno da se prilikom snimanja fotografije pazi da rijeka i njezina obala budu okomita u odnosu na dno iste. Na isti princip se otkriva i lijevi rub obale, promatranjem stupaca piksela iz sredine prema lijevo te se zapisuje njezina x koordinata.

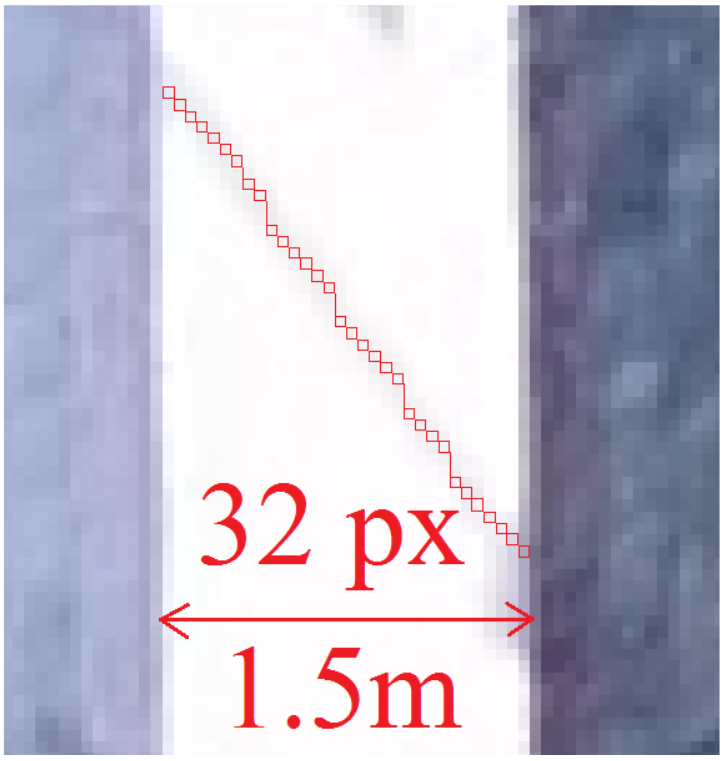
Nakon što se pronađu pozicije lijeve i desne obale, izračuna se njihova međusobna udaljenost. Kako je dobivena udaljenost (koja predstavlja širinu rijeke) izražena u pikselima, potrebno ju je izraziti u metrima. Za precizno mjerenje udaljenosti mogu se koristiti metode poput 3D *stereo vision* [17] koja zahtjeva CCD kameru ili koristeći dvije kamere s laserskom zrakom [18]. Računanje udaljenosti koristeći se jednom običnom kamerom je zahtjevno zbog toga što je potrebno znati barem dvije fizikalne veličine. Jedna od njih je udaljenost kamere od objekta kojem se uzimaju mjere, dok je druga neki referentni objekt (pokraj promatranog objekta) kojem se znaju mjere. Zato je bitno da se prilikom snimanja fotografije s drona zna kolika je njegova udaljenost od rijeke.

Kako je obala rijeke Drave projektirana i izgrađena obaloutvrdnim stilom, mogu se uzeti njezine mjere. Na slici 3.10 su prikazane mjere presjeka desne obale rijeke Drave.



Sl. 3.10. Presjek desne obale rijeke Drave

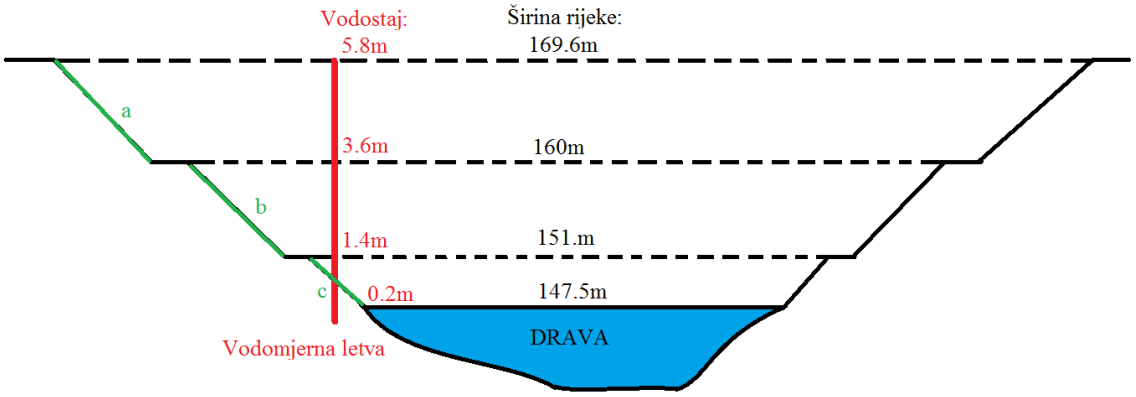
Sada kada su poznate potrebne mjere, udaljenost od rijeke i obala kao referentni objekt, može se izračunati koliko metara iznosi jedan piksel na fotografiji. Na slici 3.11 koja prikazuje jedan dio kamene obale rijeke mogu se vidjeti pikseli.



Sl. 3.11. Dio kamene obale prikazan u pikselima i metrima

Jednostavnim izračunom dobije se da jedan metar u stvarnosti sadrži približno 21.33 piksela na slici, odnosno da jedan piksel iznosi približno 0.05m. Važno je imati na umu da dobiveni omjer piksela i metra vrijedi samo ako je fotografija rijeke uslikana s udaljenosti od 112m i da ima rezoluciju i ostale karakteristike kamere poput kamere koju ima Phantom Advanced 3 dron.

Pronalaskom koordinata obale i znanjem koliko iznosi stvarna širina rijeke može se dobiti i mjera razine vode. Na slici 3.12 prikazan je presjek rijeke Drave. Dobivene mjere (sl. 3.12) izračunaju se kombinacijom slike 2.2 poznavajući koliko iznosi vodostaj na pojedinom dijelu obale, slike 3.10 gdje su poznate mjere obale i slike 3.5 na koju se primijenilo svojstvo preračunavanja piksela u metar kako bi se saznala širina rijeke za određene dijelove.



Sl. 3.12. Presjek korita Drave

Može se primijetiti da za svaku širinu rijeke postoji točno određena razina voda. Za širinu rijeke od 160m do 163m (1.5m+1.5m) vodostaj će iznositi 3.6m, za 169.6m vodostaj će biti 5.8m, itd.. Nepoznat vodostaj sa slike 3.12 je na područjima označenim zelenom bojom s a, b i c. Ako se a, b i c promatraju kao pravci definirani jednadžbama, gdje je širina rijeke x, vodostaj će iznositi y. Na slici 3.10 vidi se da se svaki pravac može posebno postaviti u koordinatni sustav s mjerama određenim pomoću slike 3.12 tako da donji kraj pravca bude u ishodištu sustava. Jednadžbu pravca bi dobili pomoću formule jednadžba pravca s dvije točke (2-1).

$$y - y_1 = \frac{y_2 - y_1}{x_2 - y_1} * (x - x_1) \qquad (2\text{-}1)$$

Pravac a bi imao točke (0,0) i (1.75, 1.2) dok bi jednadžba pravca izgledala: $y = \frac{24}{35} * x$. Kako bi se prilagodila[2] dobivena jednadžba da y govori koliko iznosi vodostaj, x se mora oduzeti sa 147,5 te ga podijeliti s 2, dok se y mora smanjiti za 0,2.

Pravac b bi imao točke (0,0) i (3.35, 2.2) dok bi jednadžba pravca izgledala: $y = \frac{44}{67} * x$. U jednadžbi se mora od x oduzeti 153,2 te ga podijeliti s 2, dok se y mora smanjiti za 1,4.

Pravac c bi imao točke (0,0) i (3.3, 2.2) dok bi jednadžba pravca izgledala: $y = \frac{2}{3} * x$. U jednadžbi se mora od x oduzeti 163 te ga podijeliti s 2, dok se y mora smanjiti za 3,6.

[2] Iz slike 3.11. može se vidjeti da donji kraj pravca počinje u točki T(147.5, 0.2) zbog toga se oduzima od x-a i zbraja y-u. Dijeli se s 2 zato što Drava ima identične obale s obje strane dok je primjer za jednu stranu. Može se točka T uvrstiti umjesto točke (0, 0) no onda ispadnu jako mali brojevi.

4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE

Aplikacija omogućuje korištenje baze podataka mobilnog uređaja na kojem se preuzmu slike dobivene s Phantom 3 Advanced drona. Aplikacija izvršava sljedeće radnje: prikaži sučelje aplikacije s dodatnim informacijama, dohvati fotografiju iz galerije mobilnog uređaja, obradi fotografiju, analiziraj rezultate dobivene obradom slike, ispiši rezultate na ekran sučelja, usporedi rezultate s prethodnim podacima, spremanje podataka u bazu i kontrola operacijskih funkcija.

4.1. Korišteni alati, tehnologije i programski jezici

Aplikacija je pisana u integriranom razvojnom okruženju Android Studio koji je specificiran za izradu Android aplikacija. Obrada slike izvršava se pomoću OpenCV library napisanog u C++ programskom jeziku. Kombinacijom dobivenih rezultata obrade i algoritma iz potpoglavlja 3.3 izrazi se vodostaj. Spajanje mobilnog uređaja s bespilotnom letjelicom izvršava se pomoću daljinskog upravljača i DJI GO aplikacije, službena aplikacija drona Phantom 3 Advanced koji se koristi kao bespilotna letjelica za dohvaćanje slika. Aplikacija DJI GO također služi za snimanje fotografija i spremanje slika (automatski) u bazu podataka. Cjelokupan kod napisan je u objektno-orijentiranom jeziku Java. Proširenu *software*-sku infrastrukturu za bespilotne letjelice može se pročitati na referenci [9].

4.1.1. Android Studio

Integrirano razvojno okruženje napravljeno u suradnji Google i JetBrains. Android Studio pruža korisničko sučelje za kreiranje aplikacija s potpuno jednostavnim pregledom za upravljanje datotekama. Omogućuje korištenje programskih jezika poput Java, Kotlin ili čak neke od takozvanih nativnih jezika poput C/C++. Android Studio daje pristup i Android SDK odnosno kompletu za razvoj *software-a* s kojim se može pristupiti potrebnim knjižnicama, pronaći pogreške u kodu, pokrenuti emulator i vidjeti kako to izgleda na mobilnom uređaju i mnoge druge stvari. Emulator zahtjeva dosta jako računalo kako bi se mogao pokrenuti. Za one sa slabijim računalima, može se spojiti vlastiti mobilni uređaj s Android Studio koji će poslužiti kao *hardware* gdje se instalira i testira aplikacija. Tijekom pokretanja programa može se ispraviti kod ako dođe do greške za koju se dobije povratna informacija i u većini slučajeva rješenje iste.

4.1.2. Dron Phantom 3 Advanced

U ovom završnom radu kao primjer bespilotne letjelice za testiranje aplikacije koristi se dron Phantom 3 Advanced tvrtke DJI prikazan na slici 4.1. Testni dron od polijetanja do slijetanja, potpuno je pod kontrolom te se pokazao iznimno stabilnim što čini let intuitivnim i vrlo jednostavnim. Jedna od prednosti ovog drona je što ima GPS i GLONASS koji mu omogućuju da bude svjestan svoje lokacije u bilo kojem trenutku. Također ima i opciju RTH (*Return to home*) pomoću koje se može vratiti dron na mjesto polijetanja ako se izgubi kontakt. Bitna stavka ove bespilotne letjelice koja zasigurno najviše utječe na rezultate aplikacije je kvaliteta kamere. Kamera ima 12 megapiksela, može snimati video 2.7K rezolucije i koristi profesionalni f/2.8 objektiv koji ima vidno polje od 94° [3].



Sl. 4.1. Prikaz izgleda Phantom 3 Advanced drona

4.1.3. DJI Development KIT

DJI tvrtka uz svoje dronove nudi i SDK razvojne alate koji omogućuju korisnicima potpunu kontrolu na svaki segment letjelice. Nude mobilni SDK koji omogućuje programerima korištenje određenih stavki i informacija u svojim aplikacijama, „Onboard SDK“ koji proširuje programske i hardverske mogućnosti drona, „Guidance SDK“ s

kojim se može pristupiti svim izlaznim podacima za glavne sustave i „Payload SDK“ koji nudi mogućnost nadogradnje drona sa senzorima ili čak i robotskim rukama [4].

4.1.4. DJI GO aplikacija

Aplikacija nudi odabair više modela DJI dronova. Odabirom na pojedini dron otvara se sučelje za kontrolu leta pomoću kojeg se može vidjeti potrebne informacije: trenutna visina drona, udaljenost od točke polijetanja, horizontalna i vertikalna brzina, postotak baterije s preostalim vremenom leta, jačina GPS signala, kvaliteta prijenosa slike i informacija koja govori o kvaliteti leta s obzirom na vremenske prilike. Također se mogu podešavati postavke upravljačkog uređaja i kamere s drona, koristiti opciju RTH i korištenje opcije polijetanja bespilotne letjelice. Aplikacija sadrži bazu podataka iz koje se mogu izvući podaci poput datuma leta, najvećom udaljenosti i visinom istog leta i vrijeme pojedinog leta. Sadrži i akademiju kroz koju se može naučiti upravljati dronom pomoću video uputa, simulatora i različitih smjernica. Također se mogu vidjeti informacije o ograničenju leta s obzirom za trenutnu poziciju drona [15].

4.1.5. OpenCV Library

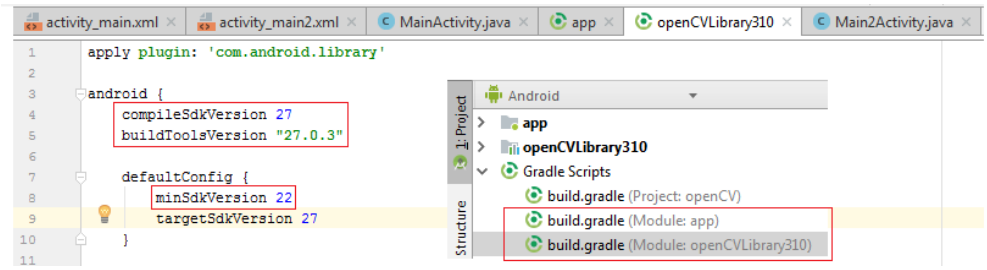
„Open Source Computer Vision Library“ je programska knjižnica čiji je izvorni kod dopušten svim programerima na uvid kako bi ga mogli koristiti, izmijeniti i nadopunjavati s vlastitim idejama. OpenCV knjižnica broji više od 2500 optimiziranih algoritama, koji uključuju sveobuhvatan skup klasičnog i najsuvremenijeg računalnog vida i algoritama strojnog učenja. Algoritmi se većinom koriste za otkrivanje i prepoznavanje lica, prepoznavanje i praćenje pokretnih objekata, praćenje kretanja putem kamere, stvaranje 3D modela i za mnoge druge stvari koje se opsežno koriste u tvrtkama, istraživački skupinama i u vladinim organizacijama [5].

4.2. Ključni dijelovi koda

U ovom poglavlju opisana je implementacija OpenCV knjižnice koja je napisana u C++ programskom jeziku, te korištenje C++ jezika u Javi. Prikazani su modeli opisani u prethodnom poglavlju pomoću programskog koda.

4.2.1. Implementacija knjižnice OpenCV u Android Studio

Sve verzije OpenCV knjižnica mogu se preuzeti s reference [5] za Windows, iOS i Android operacijski sustav s potpunom dokumentacijom i resursima. U Android Studio se implementira kao modul. Prema slici 4.2 *compileSdkVersion*, *buildToolsVersion* i *minSdkVersion* verzije moraju biti iste za modul aplikacije i modul OpenCV-a.



Sl. 4.2. Prikaz podešavanja OpenCV knjižnice.

U strukturi projekta modula aplikacije dodaje se OpenCV knjižnica kao *dependencies* za glavnu aplikaciju. Kako je OpenCV napisan u C++ jeziku, a programski kod je pisan u Javi, potrebno je napraviti generiranje nativnih datoteka koje Java razumije. To se može izvršiti pomoću NDK alata ili CMake *open-source* platforme. U ovoj aplikaciji se koristila CMake platforma s *Ninja build* sistemom za koju se mora napisati datoteka CmakeLists.txt, u kojoj se definira *path* lokacije OpenCV knjižnice i glavnog projekta, prikazana na slici 4.3.

```
set(pathToProject C:/Users/user/AndroidStudioProjects/openCV)
set(pathToOpenCV D:/OpenCV-3.1.0-android-sdk/OpenCV-android-sdk)

# For more information about using CMake with Android Studio, read the
# documentation: https://d.android.com/studio/projects/add-native-code.html

# Sets the minimum version of CMake required to build the native library.

cmake_minimum_required(VERSION 3.4.1)

set(CMAKE_VERBOSE_MAKEFILE on)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=gnu++11")

include_directories(${pathToOpenCV}/sdk/native/jni/include)

# Creates and names a library, sets it as either STATIC
# or SHARED, and provides the relative paths to its source code.
# You can define multiple libraries, and CMake builds them for you.
# Gradle automatically packages shared libraries with your APK.

add_library( # Sets the name of the library.
             native-lib

             # Sets the library as a shared library.
             SHARED

             # Provides a relative path to your source file(s).
             src/main/cpp/native-lib.cpp )

add_library( lib_opencv SHARED IMPORTED )

set_target_properties(lib_opencv PROPERTIES IMPORTED_LOCATION ${pathToProject}/app/src/main/jniLibs/${ANDROID_ABI}/libopencv_java3.so)
# Searches for a specified prebuilt library and stores the path as a
# variable. Because CMake includes system libraries in the search path by
# default, you only need to specify the name of the public NDK library
# you want to add. CMake verifies that the library exists before
# completing its build.

find_library( # Sets the name of the path variable.
              log-lib

              # Specifies the name of the NDK library that
              # you want CMake to locate.
              log )

# Specifies libraries CMake should link to your target library. You
# can link multiple libraries, such as libraries you define in this
# build script, prebuilt third-party libraries, or system libraries.

target_link_libraries( native-lib $\{log-lib} lib_opencv)
```

Sl.4.3. Prikaz CMakeLists.txt datoteka.

Nakon što se generiraju potrebne datoteke, pomoću *native-lib.cpp* (Sl. 4.4) definiraju se potrebne funkcije koristeći se *Java Native Interface* kako bi se provjerilo da je OpenCV knjižnica ispravno implementirana. Funkcije se pozivaju iz *MainActivity*-a u kojem se nalazi glavni dio koda aplikacije napisan u Javi.

```cpp
#include <iostream>
#include <jni.h>
#include <string.h>
#include <opencv2/core.hpp>

using namespace cv;
using namespace std;

extern "C" {
    jstring
    Java_semi_kruno_opencv_MainActivity_stringFromJNI(JNIEnv *env, jobject /* this */) {
        std::string hello = "Hello C++";
        return env->NewStringUTF(hello.c_str());
    }

    jstring
    Java_semi_kruno_opencv_MainActivity_validate(JNIEnv *env, jobject, jlong addrGray, jlong addrRgba) {
        cv::Rect();
        cv::Mat();
        std::string hello2 = "Hello validate";
        return env->NewStringUTF(hello2.c_str());
    }

}
```

Sl. 4.4. Prikaz *native-lib.cpp* datoteke.

Funkcije sa slike 4.4 pozivaju se metodom prikazano na slici 4.5. Kako bi *gradle* prilikom prevođenja (*compile*) znao koja je funkcija nativna, prije imena funkcije mora se dodati riječ *native.*

```
tv.setText(stringFromJNI());
if(!OpenCVLoader.initDebug()){
    tv.setText(tv.getText() + "\nOpenCVLoader ne radi.");
}
else{
    tv.setText(tv.getText() + "\nOpenCVLoader radi.");
    tv.setText(tv.getText() + "\n" + validate( matAddrGr: 0L,  matAddrRgba: 0L));
}
```

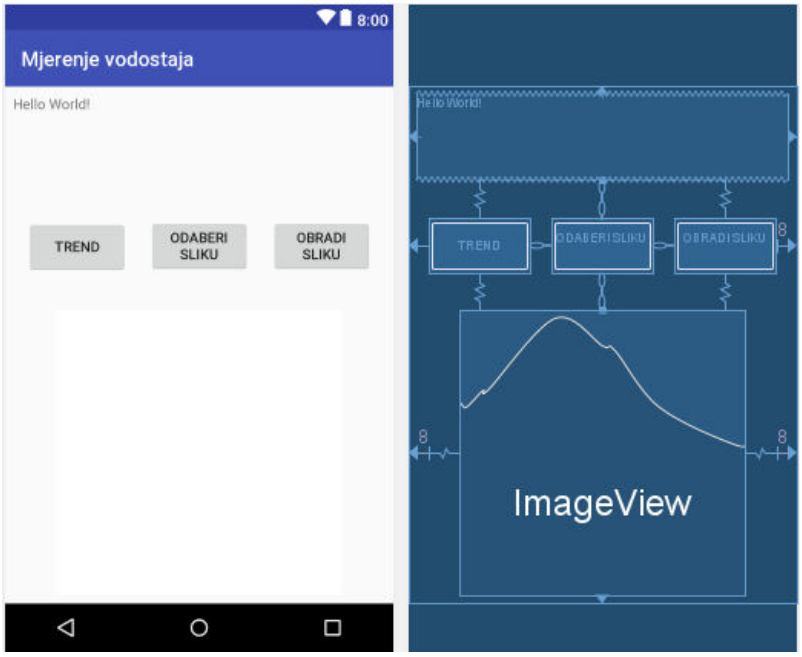
```
/**
 * A native method that is implemented by the 'native-lib' native library,
 * which is packaged with this application.
 */
public native String stringFromJNI();
public native String validate(long matAddrGr, long matAddrRgba);
```

Sl. 4.5. Pozivanje nativnih funkcija iz *native-lib.cpp.*

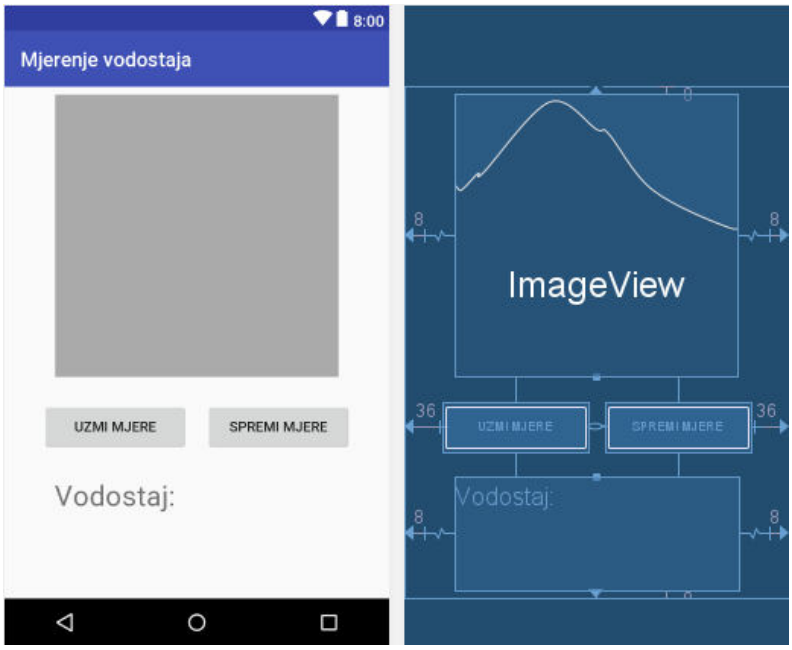
4.2.2. Prikaz izgleda sučelja aplikacije

Aplikacija se sastoji od dva sučelja: *MainActivity* i *Main2Activity*. Sučelje *MainActivity* (Sl. 4.6) se sastoji od: *textview* koji služi za prikaz teksta, tri gumba (trend, odaberi sliku i obradi sliku) i *imageview* koji služi za prikaz slika. Sučelje *Main2Activity* (Sl. 4.7) se

sastoji od *textview,* dva gumba (uzmi mjere i spremi mjere) i *imageview* za prikaz obrađene slike. U *AndroidManifest* orijentacija ekrana se postavi na portret prikaz, kako se isti ne bi okretao.



Sl. 4.6. Prikaz sučelja *MainActivity*-a.



Sl.4.7. Prikaz sučelja *Main2Activity*-a.

4.2.3. Prikaz osnovnih funkcija aplikacije

Pritiskom na gumb trend poziva se funkcija prikazana na slici 4.8 kojom se dohvaćaju podaci iz baze podataka. Podaci se redom zapisuju u *buffer* tip podataka koji se kasnije ispisuju preko *AlertDialog* na sučelje *MainActivity.* Ako nema podataka u bazi podataka, korisniku će se na ekran pojaviti obavijest: „Nema podataka u bazi podataka“.

```
private void otvoriTrend(){
    Cursor res = moja_baza.getData();
    if(res.getCount()==0)
        Toast.makeText( context: MainActivity.this, text: "Nema podataka u bazi",Toast.LENGTH_LONG).show();
    else{
        StringBuffer buffer=new StringBuffer();
        while(res.moveToNext()){
            buffer.append("ID: "+res.getString( columnIndex: 0)+"\n");
            buffer.append("Datum: "+res.getString( columnIndex: 1)+"\n");
            buffer.append("Vodostaj: "+res.getString( columnIndex: 2)+"m\n");
            buffer.append("Širina rijeke: "+res.getString( columnIndex: 3)+"m\n");
            buffer.append("Trend: "+res.getString( columnIndex: 4)+"cm\n\n");
        }
        prikaziBazu( naslov: "Trend vodostaja", buffer.toString());
    }
}
public void prikaziBazu(String naslov, String poruka){
    AlertDialog.Builder builder=new AlertDialog.Builder( context: this);
    builder.setCancelable(true);
    builder.setTitle(naslov);
    builder.setMessage(poruka);
    builder.show();
}
```

Sl. 4.8. Prikaz funkcije trenda.

Odabir slike (Sl. 4.9) iz memorije mobilnog uređaja vrši se preko *Intent* kojim se aktivira otvaranje galerije mobilnog uređaja. Odabirom slike sprema se njezin *Uri* (lokacija slike na mobilnom uređaju, slično *URL* web stranice). Korištenjem *ExifInterface* klase, koja služi za čitanje i pisanje oznaka na slici (npr. čitanje pojedinosti slike ili određivanje orijentacije slike), spremi se datum nastanka slike. Ako se ne može pročitati datum slike, spremit će se u bazu podataka obavijest u obliku teksta: „nema“.

```
private void otvoriGaleriju(){
    Intent Galerija =new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.INTERNAL_CONTENT_URI);
    startActivityForResult(Galerija, PICK_IMAGE);
}
protected void onActivityResult(int RequestCode, int ResultCode, Intent data){
    super.onActivityResult(RequestCode, ResultCode, data);
    String datetext="nema";
    if(ResultCode == RESULT_OK && RequestCode == PICK_IMAGE){
        ImageUri =data.getData();
        Slika.setImageURI(ImageUri);
        try {
            InputStream in;
            in=getContentResolver().openInputStream(ImageUri);
            ExifInterface intf = new ExifInterface(in);
            datetext=intf.getAttribute(ExifInterface.TAG_DATETIME);
        } catch(IOException e) {
            e.printStackTrace();
        }
        datum=datetext;
    }
```

Sl. 4.9. Prikaz funkcije odabira fotografije.

Pritiskom na gumb obradi sliku (Sl. 4.10) poziva se novi *Intent* pomoću kojeg se šalje *Uri* fotografije (ako se prethodno izabrala slika) i otvara *Main2Activity.*

```
private void obradiSliku() {
    Intent otvoriNoviProzor = new Intent( packageContext: MainActivity.this, Main2Activity.class);
    if (ImageUri == null) {
        Toast.makeText( context: MainActivity.this,  text: "Izaberi prvo sliku", Toast.LENGTH_LONG).show();
    } else {
        otvoriNoviProzor.putExtra( name: "URIslike", ImageUri.toString());
        startActivityForResult(otvoriNoviProzor,  requestCode: 1);
    }
}
```

Sl. 4.10. Prikaz funkcije obradi sliku.

Otvaranjem *Main2Activity* (Sl. 4.11) kreira se bitmap slika dobivena preko *Uri* iz *MainActivity,* te se postavlja njezina konfiguracija, ako je nema, na format *ARGB_8888* (svaki piksel je pohranjen na 4 bajta) koju podržava OpenCV knjižnica. Također, kreira se i *Canvas* kako bi se omogućilo korištenje opcije crtanja za označavanje obala rijeke.

```
Bundle slika = getIntent().getExtras();
if(slika!=null){
    String obradjena_URI = slika.getString( key: "URIslike");
    ImageUri=Uri.parse(obradjena_URI);
    try {
        tempbitmap = BitmapFactory.decodeStream(getContentResolver().openInputStream(ImageUri));
        Bitmap.Config config;
        if(tempbitmap.getConfig() != null){
            config = tempbitmap.getConfig();
        }else{
            config = Bitmap.Config.ARGB_8888;
        }

        bitmap = Bitmap.createBitmap(
                tempbitmap.getWidth(),
                tempbitmap.getHeight(),
                config);

        Canvas canvasMaster = new Canvas(bitmap);
        canvasMaster.drawBitmap(tempbitmap,  left: 0,  top: 0,  paint: null);

        obradjena_slika.setImageBitmap(bitmap);
        crno_bijela();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
```

Sl. 4.11. Kreiranje bitmap i *Canvas.*

Kao što je objašnjen redoslijed izvršavanja funkcija za obradu slike (dijagram na slici 3.7), tako se izvršava i u aplikaciji. *Utils* klasa iz OpenCV knjižnice sadrži metodu *bitmapToMat* koja pretvara bitmap format slike u *Mat* oblik. Nakon toga izvršavaju se funkcije *cvtColor* (treba navesti u koji se oblik želi pretvoriti slika, u ovom slučaju sliku sa sivim tonovima), *blur* (treba izabrati veličinu jezgre, u ovom slučaju 3x3) i *Canny* (izabere se veličina jezgre, vrijednost donjeg i gornjeg praga) iz klase *Imgproc* te se dobiveni rezultati (Sl. 4.12) prikazuju u *Imageview* na sučelju *Main2Activity.*

```
public void crno_bijela(){
    for(int i=0;i<=5;i++) { //zbog bolje uočljivosti ruba rijeke
        Mat image = new Mat();
        Mat grayImage = new Mat();
        Mat detectedEdges = new Mat();
        Utils.bitmapToMat(bitmap, image);
        Imgproc.cvtColor(image, grayImage, Imgproc.COLOR_BGR2GRAY);
        Imgproc.blur(grayImage, detectedEdges, new Size( width: 3,  height: 3));
        Imgproc.Canny(detectedEdges, detectedEdges,  threshold1: 100,  threshold2: 50,  apertureSize: 3,  L2gradient: false);
        Utils.matToBitmap(detectedEdges, bitmap);
    }
}
```

Sl. 4.12. Prikaz funkcija *cvtColor, blur* i *Canny.*

Pritiskom na gumb uzmi mjere izvršavaju se četiri funkcije: pronađi lijevu i desnu obalu (Sl. 4.13), pronađi širinu rijeke (Sl. 4.14) i pronađi vodostaj (Sl. 4.15). Bitmap slika u aplikaciji je prikazana kao rešetka piksela sa stupcima i redcima. Traženje lijeve i desne obale izvršava se for petljama (jedna je za stupac, druga za redak). Kada se u stupcu izbroji više od 700 bijelih piksela (vrijednost bijelog piksela u RGB: crvena=255, zelena=255 i plava=255), on se označi sa zelenom crtom, spremi se broj stupca i petlje se prekidaju. Širina rijeke se dobije tako što se oduzmu vrijednosti pronađenih stupaca te ih se podjeli s vrijednosti 21,33 (objašnjeno u 3.3.1). Vrijednost varijable širine rijeke nakon toga prolazi kroz if slučajeve iz kojih se na sučelje ispisuje vrijednost vodostaja.

```
public void pronadjiDesnuObalu(){
    for(int x = bitmap.getWidth()/2; x < bitmap.getWidth(); x++) {
        int brojac=0;
        for (int y = 0; y < bitmap.getHeight(); y++) {
            int piksel = bitmap.getPixel(x, y);
            if(Color.red(piksel)==255 && Color.blue(piksel)==255 && Color.green(piksel)==255){
                brojac++;
                if(brojac>700){
                    Canvas canvas=new Canvas(bitmap);
                    Paint zelena=new Paint();
                    zelena.setColor(Color.GREEN);
                    zelena.setStrokeWidth(15);
                    canvas.drawLine(x, startY: 0,x, bitmap.getHeight(),zelena);
                    x1=x; //x koordinata desne obale
                    obradjena_slika.setImageBitmap(bitmap);
                }
            }
        }
        if(brojac>700) {
            break;
        }
    }
}
public void pronadjiLijevuObalu(){
    for(int x = bitmap.getWidth()/2; x >= 0; x--) {
        int brojac=0;
        for (int y = 0; y < bitmap.getHeight(); y++) {
            int piksel = bitmap.getPixel(x, y);
            if(Color.red(piksel)==255 && Color.blue(piksel)==255 && Color.green(piksel)==255){
                brojac++;
                if(brojac>700){
                    Canvas canvas=new Canvas(bitmap);
                    Paint zelena=new Paint();
                    zelena.setColor(Color.GREEN);
                    zelena.setStrokeWidth(15);
                    canvas.drawLine(x, startY: 0,x, bitmap.getHeight(),zelena);
                    x2=x; //x koordinata lijeve obale
                    obradjena_slika.setImageBitmap(bitmap);
                }
            }
        }
        if(brojac>700) {
            break;
        }
    }
}
```

Sl. 4.13. Prikaz funkcija pomoću kojih se pronalazi lijeva i desna obala rijeke.

```
public void pronadjiSirinuRijeke(){
    PixelsirinaRijeke=x1-x2;
    MetarsirinaRijeke=PixelsirinaRijeke/21.3333333333333;
} // 1 metar u stvarnosti iznosi 21.3333333333333 pixela na slici rezolucije 4000x3000
```

Sl. 4.14. Prikaz funkcije pomoću koje se uzima širina rijeke.

```
public void pronadjiVodostaj(){
    double x;
    double y;
    if(MetarsirinaRijeke==0){
        vodostaj_text.setText(("Greška!!!"));
    }
    if(0<MetarsirinaRijeke && MetarsirinaRijeke<147.5){ //0.etapa
        y=0.08*MetarsirinaRijeke-11.6;
        vodostaj_text.setText("Vodostaj: "+String.valueOf(zaokruzi(y))+"m\nŠirina rijeke: "+ String.valueOf(zaokruzi(MetarsirinaRijeke))+"m");
        Vodostaj_mjera=zaokruzi(y);
    }
    if(147.5<=MetarsirinaRijeke && MetarsirinaRijeke<151) { //1.etapa
        x = MetarsirinaRijeke - 147.5;
        x = x / 2;
        y = (24 / 35) * x;
        y = y + 0.2;
        vodostaj_text.setText("Vodostaj: " + String.valueOf(zaokruzi(y)) + "m\nŠirina rijeke: " + String.valueOf(zaokruzi(MetarsirinaRijeke))+"m");
        Vodostaj_mjera=zaokruzi(y);
    }
    if(151<=MetarsirinaRijeke && MetarsirinaRijeke<153.2){ //<1., 2.>etapa
        vodostaj_text.setText("Vodostaj: 1.4m\nŠirina rijeke: " + String.valueOf(zaokruzi(MetarsirinaRijeke))+"m");
        Vodostaj_mjera=1.40;
    }
    if(153.2<=MetarsirinaRijeke && MetarsirinaRijeke<160) { //2.etapa
        x = MetarsirinaRijeke - 153.2;
        x = x / 2;
        y = (44 / 67) * x;
        y = y + 1.2 + 0.2;
        vodostaj_text.setText("Vodostaj: " + String.valueOf(zaokruzi(y)) + "m\nŠirina rijeke: " + String.valueOf(zaokruzi(MetarsirinaRijeke))+"m");
        Vodostaj_mjera=zaokruzi(y);
    }
    if(160<=MetarsirinaRijeke && MetarsirinaRijeke<163) { //<2., 3.>etapa
        vodostaj_text.setText("Vodostaj: 3.6m\nŠirina rijeke: " + String.valueOf(zaokruzi(MetarsirinaRijeke))+"m");
        Vodostaj_mjera=3.60;
    }
    if(163<=MetarsirinaRijeke && MetarsirinaRijeke<169.6) { //3.etapa
        x = MetarsirinaRijeke - 163;
        x = x / 2;
        y = (2 / 3) * x;
        y = y + 3.4 + 0.2;
        vodostaj_text.setText("Vodostaj: " + String.valueOf(zaokruzi(y)) + "m\nŠirina rijeke: " + String.valueOf(zaokruzi(MetarsirinaRijeke))+"m");
        Vodostaj_mjera=zaokruzi(y);
    }
    if(MetarsirinaRijeke>=169.6){
        vodostaj_text.setText(("Vodostaj: >5.8m\nŠirina rijeke: " + String.valueOf(zaokruzi(MetarsirinaRijeke))+"m"));
        Vodostaj_mjera=5.80;
    }
}
```

Sl. 4.15. Prikaz funkcije pomoću koje se uzima mjera vodostaj.

Pritiskom na gumb spremi mjere (Sl. 4.16) poziva se *Intent* pomoću kojeg se šalju mjere (ako postoje) vodostaja i širine rijeke u *MainActivity* te se zatvara *Main2Activity*.

```
public void spremiMjerenja(){
    if(Vodostaj_mjera!= 0 && MetarsirinaRijeke!=0) {
        Intent pocetni_prozor = new Intent();
        pocetni_prozor.putExtra( name: "vodostaj", String.valueOf(Vodostaj_mjera));
        pocetni_prozor.putExtra( name: "sirina", String.valueOf(zaokruzi(MetarsirinaRijeke)));
        setResult(RESULT_OK, pocetni_prozor);
        finish();
    }
    else
        Toast.makeText( context: Main2Activity.this, text: "Prvo uzmi mjere rijeke!", Toast.LENGTH_LONG).show();
}
```

Sl. 4.16. Prikaz funkcije spremi mjere.

Zatvaranjem *Main2Activity,* računa se trend vodostaja u odnosu na zadnje mjerenje. Ako nema zadnjeg mjerenja, za trend se upisuje vrijednost „+0“. Funkcijom ubaci podatke (Sl. 4.17) spremaju se vrijednosti mjere vodostaja, širine rijeke, datum uslikane slike i trend.

```
if(ResultCode==RESULT_OK && RequestCode==1){
    vodostaj=Double.parseDouble(data.getStringExtra( name: "vodostaj"));
    sirina=Double.parseDouble(data.getStringExtra( name: "sirina"));

    Cursor res=moja_baza.getData();
    double zadnje_mjerenje;
    int trend_value;

    if(res.getCount()==0){
        ubaci_podatke( vrijednost_trenda: 0,  predznak: "+");
    }
    else{
        res.moveToLast();
        zadnje_mjerenje=res.getDouble( columnIndex: 2);
        zadnje_mjerenje=zadnje_mjerenje*100; //kako bi dobili zadnji zapisan vodostaj u cm
        vodostaj=vodostaj*100; //kako bi dobili trenutni vodostaj u cm
        trend_value=(int)(vodostaj-zadnje_mjerenje);
        vodostaj=vodostaj/100; //kako bi vratili trenutni vodostaj u m
        if(trend_value>=0) {
            ubaci_podatke(trend_value,  predznak: "+");
        }
        else{
            ubaci_podatke(trend_value,  predznak: ""); //predznak ce biti "-"
        }
    }

}
public void ubaci_podatke(int vrijednost_trenda, String predznak){
    boolean provjera= moja_baza.insertData(datum,vodostaj,sirina, trend: predznak+String.valueOf(vrijednost_trenda));
    if(provjera==true)
        Toast.makeText( context: MainActivity.this, text: "Uspješno spremljeno",Toast.LENGTH_LONG).show();
    else
        Toast.makeText( context: MainActivity.this, text: "Greška u spremanju",Toast.LENGTH_LONG).show();
}
```

Sl. 4.17. Prikaz spremanja vrijednosti u bazu podataka.

4.2.4. Prikaz korištenih paketa u aplikaciji

Kako bi programski kod obavljao prethodno navedene funkcije, potrebno je uvesti sljedeće pakete prikazane na slici 4.18 za *MainActivity* te na slici 4.19 za *Main2Activity.*

```
import android.app.AlertDialog;
import android.content.Intent;
import android.database.Cursor;
import android.support.media.ExifInterface;
import android.net.Uri;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import org.opencv.android.OpenCVLoader;
import java.io.IOException;
import java.io.InputStream;
```

Sl. 4.18. Prikaz programskih paketa korištenih u *MainActivity.*

```
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import org.opencv.android.Utils;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
```

Sl. 4.19. Prikaz programskih paketa korištenih u *Main2Activity.*

4.3. Programska arhitektura

Programske arhitekture uglavnom se koriste kako bi poboljšali rad same aplikacije. Postoji nekoliko obrasca koji imaju svrhu odvojiti dijelove programskog koda aplikacije u komponente. Najpoznatiji i najrašireniji je MVC (*Model-View-Controller*) [19] obrazac. Sastoji se od: *Modela* koji upravlja podacima i osnovnom logikom aplikacije, *Pogleda* koji ima svrhu prikaza podataka na sučelje i *Upravitelj* koji izvršava korisnikove radnje i zahtjeve. Sličnu arhitektura ima i aplikacija koju opisuje završni rad. Model koji upravlja bazom podataka i njezinim podacima, jedan oblik pogleda koji prikazuje rezultate na sučelje i upravitelj koji kontrolira i izvršava korisnikove radnje i očekivanja.

4.4. Prikupljanje i spremanje podataka

Slike dobivene s bespilotne letjelice Phantom 3 Advanced preuzimaju se u punoj rezoluciji te se spremaju u bazu mobilnog uređaja. Nakon toga obradom dobivene fotografije spremaju se podaci u SQLite bazu podataka implementiranu zajedno s aplikacijom. Baza podataka se sastoji od atributa: ID tipa *integer* koji predstavlja primarni ključ baze, DATUM tipa *text,* VODOSTAJ tipa *double,* ŠIRINA tipa *double* i TREND tipa *varchar*. Sljedeći kod (Sl. 4.20) prikazuje kako se baza podataka kreira i briše.

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE "+TABLE_NAME+"(ID INTEGER PRIMARY KEY, DATUM TEXT, VODOSTAJ DOUBLE, ŠIRINA DOUBLE, TREND VARCHAR(7));");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
    onCreate(db);
}
```

Sl. 4.20. Prikaz kreiranja i brisanja SQLite baze podataka.

Funkcije dohvaćanje (izvodi se preko upita gdje su izabrani svi podaci na uvid iz spremljene baze podataka) i spremanje (pomoću klase *ContentValues* ubace se vrijednosti u bazu podataka) podataka izvodi se kodom prikazanim na slici 4.21.

```
public boolean insertData(String datum, double vodostaj, double sirina, String trend){
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues contentValues=new ContentValues();
    contentValues.put(COL_DATUM, datum);
    contentValues.put(COL_VODOSTAJ, vodostaj);
    contentValues.put(COL_SIRINA, sirina);
    contentValues.put(COL_TREND, trend);
    long result = db.insert(TABLE_NAME,  nullColumnHack: null, contentValues);
    if(result==-1)
        return false;
    else
        return true;
}

public Cursor getData(){
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor res=db.rawQuery( sql: "SELECT * FROM "+TABLE_NAME, selectionArgs: null);
    return res;
}
```

Sl. 4.21. Prikaz dohvaćanja i spremanja podataka u SQLite bazu podataka.

5. NAČIN RADA I KORIŠTENJE APLIKACIJE

U ovom poglavlju prikazano je korištenje aplikacije i njezine mogućnosti, primjer rada aplikacije na konkretnom primjeru prilikom obrađivanja slike rijeke te analiza cjelokupnog rada aplikacije s prikazanim mogućnostima poboljšanja.

5.1. Korištenje aplikacije

Pritiskom na ikonu aplikacije koja se nalazi na mobilnom uređaju otvara se sučelje prikazano na slici 5.1.

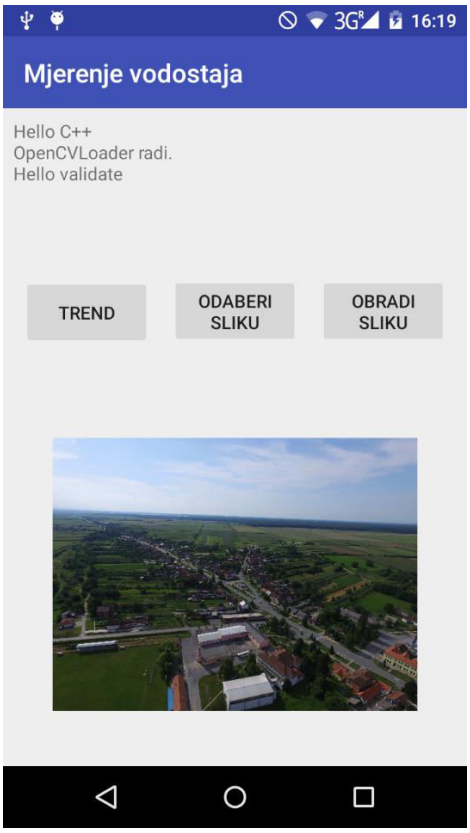


Sl. 5.1. Prikaz sučelja aplikacije.

Može se uočiti da knjižnica OpenCV radi te da je moguća obrada slike. Pritiskom na tipku trend otvara se *AlertDialog* (Sl. 5.2) na kojemu se ispisuju podaci iz baze podataka aplikacije, odnosno dobije se na uvid za sve spremljene datume koliko iznosi vodostaj i širina rijeka te iznos trenda u odnosnu na prethodni datum. Pritiskom na odaberi sliku otvara se galerija mobilnog uređaja (moguć je odabir samo fotografija, ne i video zapisa) i odabirom na željenu fotografiju otvara se u obliku bitmap slike na sučelju aplikacije kao što je prikazano na slici 5.3.



Sl. 5.2. Prikaz baze podataka na sučelje aplikacije.



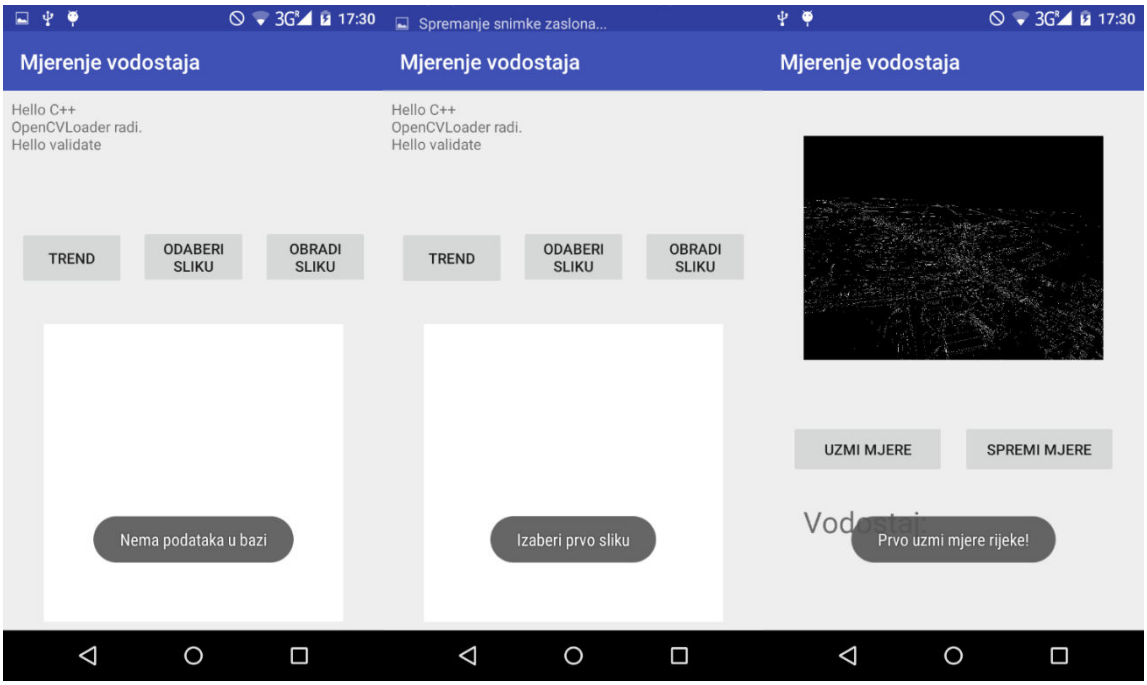
Sl. 5.3. Prikaz odabrane slike na sučelje aplikacije.

Pritiskom na obradi sliku otvara se novi prikaz (Sl. 5.4) na sučelju s prethodnom obrađenom (metodom opisanom u poglavlju 3.3) odabranom slikom iz galerije. Odabirom uzmi mjere, aplikacija izvršava funkcije pronađi lijevu i desnu obalu. Ako je ne nađe, aplikacija će ispisati na sučelje „Greška!!!“ kao što je prikazano na slici 5.4.



Sl. 5.4. Prikaz obrađene slike na sučelje.

U aplikaciji također postoji kontrola odabira funkcije. Ne može se otvoriti baza podataka ako nema podataka u istoj. Također ne može se obraditi slika ukoliko se nije prethodno

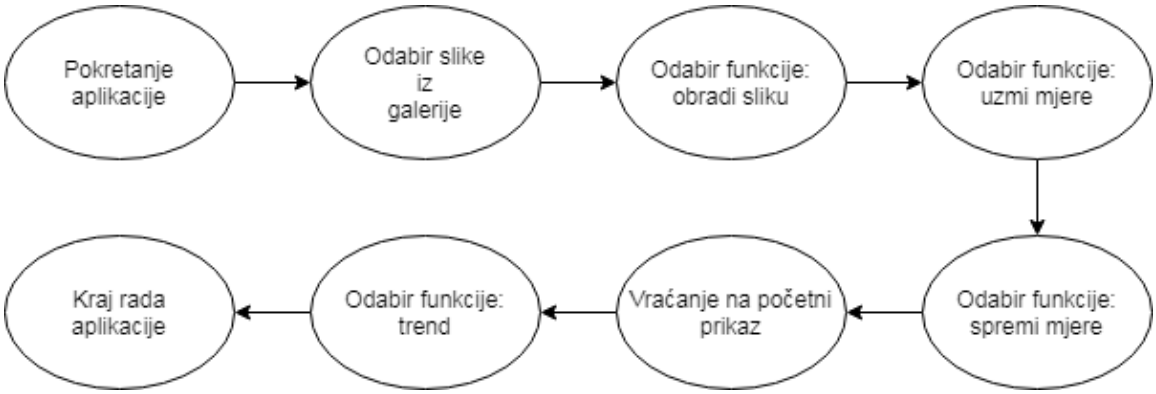


odabrala slika. Na isti princip ne mogu se spremiti mjere ako prethodno nisu izračunate. Aplikacija ima svoj redoslijed izvršavanja funkcija kako ne bi došlo do pogreške i pada aplikacije. Korisnik dobije obavijest ako dođe do odabira neželjene radnje te mu aplikacija istu onemogućuje kao što je prikazano na slici 5.5. Osim kontrole odabira funkcija, prilikom spremanja podataka u bazu podataka, aplikacija također daje poruku korisniku „Uspješno spremljeno“ ako su se podaci spremili ili „Greška u spremanju“ ako se nisu spremili. Za mjere kao što su vodostaj, širina rijeke i trend, aplikacija ima mogućnost spremanja u bazu podataka ukoliko se dobiju vrijednosti za njih. Mjera datum je zasebna te se ona dobije prilikom odabira slike iz galerije. Ako se ne može pročitati datum nastanka slike, aplikacija će ispisati za tu vrijednost „nema“ u bazu podataka.

Sl. 5.5. Prikaz kontrole aplikacije prilikom odabira funkcija.

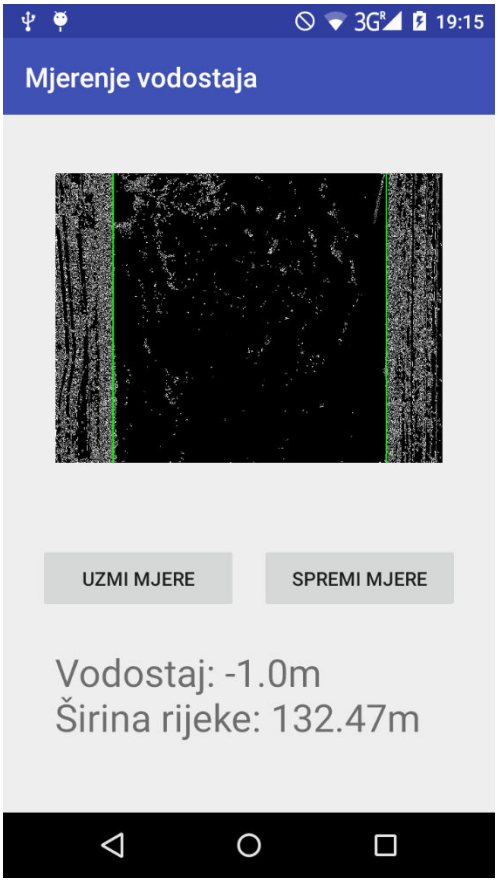
5.2. Prikaz primjera obrade slike u aplikaciji

Slika 5.6 prikazuje dijagram koji opisuje pravilan redoslijed izvršavanja funkcija kako bi se odabranom slikom iz galerije dobio vodostaj i spremio u bazu podataka.



Sl. 5.6. Prikaz pravilnog izvršavanja funkcija u aplikaciji.

Ulaskom u aplikaciju izabere se opcija „odaberi sliku“ te se izabere slika (Sl. 3.5) iz galerije. Nakon toga se odabere funkcija obradi sliku. Aplikacija obradi sliku metodom opisanom u 3. poglavlju i prikaže ju na sučelju (Sl. 5.7). Pritiskom na „uzmi mjere“ aplikacija pronađe obale rijeke te ih označi na slici rijeke, pritom na zaslon sučelja ispisuje mjeru razine vode i trenutnu širinu rijeke kao što je prikazano na slici 5.7.

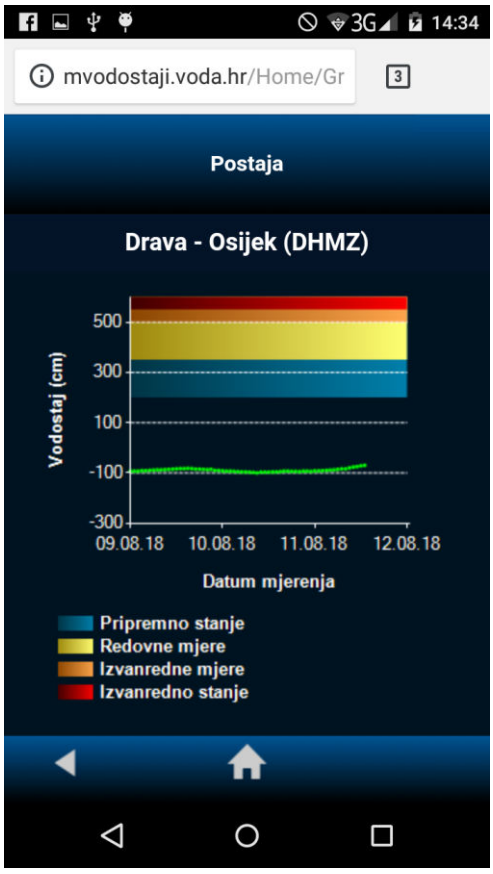


Sl. 5.7. Prikaz rezultata obrade slike u aplikaciji.

Pritiskom na tipku „spremi mjere“ vodostaj se sprema u bazu podataka zajedno s prikazanom širinom rijeke, trendom porasta/pada od zadnjeg mjerenja te datum uslikane slike.

5.3. Analiza rada sustava

Na slici 5.2 vide se podaci pod „ID: 1“ za sliku 3.5 koja je ispravno uslikana i pod „ID: 2“ za sliku 3.6. koja je nepravilno uslikana. Obje slike su uslikane istog datuma s vremenskom razlikom od 46 sekundi, a može se primijetiti da se razina vode između te dvije slike znatno razlikuje. Vodostaj za „ID: 2“ je pola metra manji od vodostaja „ID: 1“. Rad ove aplikacije je osjetljiv te treba pripaziti i pravilno snimiti slike kako je opisano u 3. poglavlju. Ako je zakrivljenost slike veća, vodostaj će za sustav biti manji. Na slici 5.8 može se primijetiti kako se kretao vodostaj tih dana (nemogućnost prikaza vodostaja za datum 06.08. no može se primijetiti da se kretao oko -1m) izmjeren mjernim stanicama Hrvatskih voda [2].



Sl. 5.8. Prikaz vodostaja za vremenski interval 09.08. do 12.08. [2].

Tablica 5.1 pokazuje usporedbu mjerenja vodostaja koja su se izvršila s dronom i stvarna mjerenja vodostaja izmjerena na mjernim stanicama.

Tab. 5.1. Prikaz vodostaja izmjeren dronom i na mjernim stanicama

Datum	Vodostaj-dron	Vodostaj-stvarni
06.08.2018.	-1m	približno -1m
12.08.2018.	-0,96m	-0,91m
17.08.2018.	-0,79m	-0,83m
20.08.2018.	-0,92m	-0,88m

Može se primijeti da se vodostaj aplikacije koju predstavlja ovaj završni rad i vodostaj aplikacije koji je izmjeren na mjernim stanicama na Dravi u Osijeku podudara s eventualnom malom pogreškom. Vodostaj Drave tijekom ljetnih mjeseci slabo se mijenja. Za potpunu provjeru ispravnosti za bilo koju vrijednost vodostaja treba veći vremenski period testiranja (koji može potrajati i par godina), zbog toga se ne može u potpunosti analizirati rad aplikacije. Nadalje, zbog fleksibilnosti i mogućnosti poboljšanja algoritma za računanje vodostaja (prikazanog u 3. poglavlju), moguće je mjerenje vodostaja pomoću bespilotne letjelice i obradom dobivene slike. Pogreške se mogu ispraviti na sljedeće načine: ispravljanje jednadžbe pravaca a, b i c (opisani u poglavlju 3.3.1), dijeljenje a, b i c pravca na manje pravce s vlastitim jednadžbama ili napraviti tablicu (metodom sličnoj metodi uzorkovanje [20]) pomoću koje bi za svaku vrijednost širine rijeke postojala vrijednost vodostaja.

6. ZAKLJUČAK

U ovom radu je modelirana i razvijena mobilna Android aplikacija za praćenje vodostaja rijeke Drave temeljena na obradi i analizi slike snimljene bespilotnom letjelicom Phantom 3 Advanced. Knjižnica OpenCV i njene metode za filtriranje, uklanjanje smetnji i detekciju rubova su poslužile u obradi slike. Obrađena slika se korištenjem algoritma za detekciju rubnih dijelova između rijeke i obala te njihove međusobne udaljenosti preračunava u vodostaj koji se tada prikazuje korisniku aplikacije na uvid uz mogućnost spremanja u bazu podataka. Korisnik se može poslužiti bazom spremljenih podataka kako bi uvidio trend rasta/pada vodostaja. Ova aplikacija povezuje dva uređaja - pametni telefon i dron, a služi kako bi se dobila visina vodostaja na vrlo učinkovit i jednostavan način u slučaju zatajenja mjernih stanica. Aplikacija je pisana u programskom jeziku Java na integriranom razvojnom okruženju Android Studio, a ispitana je na primjerima podataka rijeke Drave. Navedenim se podacima pokazalo kako se vodostaj aplikacije podudara sa stvarnim vrijednostima. Iako su rezultati obrade obećavajući, ispitivanje treba izvršiti i za druge vrijednosti razine vode. Ako rezultati ne budu točni, sustav je prilagodljiv te je na njemu moguće izvršiti poboljšanja. Ovakav sustav mjerenja vodostaja obradom slike s bespilotne letjelice moguće je primijeniti i na druge rijeke izradom detaljnog presjeka obale rijeke.

LITERATURA

[1] S. O'Donnell, A Short History Of Unmnned Aerial Vehicles, Consortiq, 2017., dostupno na: https://consortiq.com/en-gb/media-centre/blog/short-history-unmanned-aerial-vehicles-uavs [20. lipanj 2018.]

[2] Hrvatske Vode, Vodostaji-Hrvatske Vode, Google Play, 2018., dostupno na: https://play.google.com/store/apps/details?id=com.hrvatskevode [23. lipanj 2018.]

[3] DJI, Phantom 3 Advanced, DJI, 2018., dostupno na: https://www.dji.com/phantom-3-adv [24. lipanj 2018.]

[4] DJI, DJI SDK, Developer DJI, 2018., dostupno na: https://developer.dji.com/ [24. lipanj 2018.]

[5] OpenCV, About OpenCV, OpenCV, 2018., dostupno na: https://opencv.org/about.html [24. lipanj 2018.]

[6] S. Marzukhi, M. Amirul Shafiq Mohamad Sidik, H. Mohd Nasir, Z. Zainol, M. Nazri Ismail, Flood Detection And Warning System (Flows), Proceedings Of The 12th International Conference On Ubiquitous Information Management And Communication, Incom ’18, Article No. 36, Langkawi, Malaysia, 2018.

[7] R. Nishikawa, N. Thepvilojanapong, N. Kitsunezaki, Y. Tobe, Planning-Based Routing For Area Sensing, Proceedings Of The 2014 International Workshop On Web Intelligence And Smart Sensing, pp 1-2, Saint Etienne, France, 2014.

[8] A. Mancini, E. Frontoni, P. Zingaretti, S. Longhi, High-Resolution Mapping Of River And Estuary Areas By Using Unmanned Aerial And Surface Platform, International Conference On Unmanned Aircraft Systems (Icuas), Denver, Colorado, USA, 2015.

[9] Y. David Liu, L. Ziarek, Toward A Java Based Infrastructure For Unmanned Aerial Vehicles, Companion Proceedings Of The 2015 Acm Sigplan International Conference On Systems, Programming, Languages And Applications: Software For Humanity, Splash Companion 2015, pp 56-57, Pittsburgh, PA, USA, 2015.

[10] Cheng-Yuan Li, Hao-Hua Chu, Blimpprobe: An Aerial Surveillance Platform, Proceedings Of The 11th International Conference On Information Processing In Sensor Networks, IPSN ’12, pp 131-132, Beijing, China, 2012.

[11] J. Ondraček, O. Vanek, M. Pechouček, Solving Infrastructure Monitoring Problems With Multiple Heterogeneous Unmanned Aerial Vehicles, Proceedings Of The 2015 International Conference On Autonomous Agents And Multiagent Systems, AAMAS ’15, pp 1597-1605, Instabul, Turkey, 2015.

[12] E. Honkavaara, H. Saari, J. Kaivosoja, I. Polonen, T. Hakala, P. Litkey, J. Makynen, L. Pesonen, Processing And Assessment Of Spcetrometric, Stereoscopic Imagery Collected Using A Lightweight UAV Spectral Camera For Precision Agriculture, Remote Sensing, 2013., dostupno na: http://www.mdpi.com/2072-4292/5/10/5006 [2. Kolovoz 2018.]

[13] Hrvatske Vode, Karte Opasnosti Od Poplava I Karte Rizika Od Poplava, Hrvatske Vode, 2014., dostupno na: http://korp.voda.hr/ [4. Kolovoz 2018.]

[14] M. Wozniak, G. Qu, DEVS-Flood: A Cellular DEVS Approach To Flood Inundation Modeling (WIP), Proceedings Of The 2012 Symposium On Theory Od Modeling And Simulation, TMS/DEVS ’12, Article No. 53, Orlando, Florida, 2012.

[15] DJI Technology CO.,LTD, DJI GO—For Products Before P4, Google Play, 2018., dostupno na: https://play.google.com/store/apps/details?id=dji.pilot [20. Srpanj 2018.]

[16] Y. Li, L. Liu, L. Wang, D. Li, M. Zhang, Fast SIFT Algorithm Based On Sobel Edge Detector, 2nd International Conference On Consumer Electronics, Communications And Networks (Cecnet), pp 1820-1823, Yichang, China, 2012.

[17] J. J. Aguilar, F. Torres, M. A. Lope, Stereo Vision For 3D Measurement: Accuracy Analysis, Calibration And Industrial Applications, Measurement, vol. 18, pp 193-200, August 1996.

[18] J. L. Vilaca, J. C. Fonseca, A. M. Pinho, Calibration Procedure For 3D Measurement System Using Two Cameras And Laser Line, Optics And Laser Technology, vol. 41, pp 112-119, March 2009.

[19] C. Liyan, Application Reserach Of Using Design Pattern To Improve Layered Architecture, 2009 IITA International Conference On Control, Automation And Systems Engineering, pp 303-306 Zhangjiajie, China, 2009.

[20] C. F. N. Cowan, Digital Processing Of Signals – Theory And Practice, IEE Proceedings F – Commnuications, Radar And Signal Processing, vol. 132, pp 202-203, June 1985.

SAŽETAK

Bespilotne letjelice poput dronova postaju sve popularnije u svijetu tehnologija upravo zbog toga što mogu „vidjeti“ iz perspektiva iz kojih ljudi ne mogu. Osim što se koriste u komercijalne i vojne svrhe, također se mogu primjenjivati i u istraživačkim radovima. U ovom završnom radu prikazana je upotreba Phantom 3 Advanced drona u svrhu mjerenja vodostaja rijeke Drave i to obradom slika u Android aplikaciji koristeći biblioteku OpenCV, te pomoću razrađenog algoritma (u ovom radu) na temelju karakteristika i vlastitog mjerenja obale rijeke Drave izračuna vodostaja. Osim toga, prikazane su sve mogućnosti drona koristeći se DJI GO aplikacijom, te njegovog mogućeg proširenja i poboljšanja pomoću DJI Development KIT alata. Opisan je cijeli proces od snimanja, spremanja i obrađivanja slika, do izgleda sučelja Android mobilne aplikacije i prikaza dobivenih rezultata na istom. Korištenje aplikacije je intuitivno i funkcijski ograničeno. Rezultati analize pokazuju obećavajuće rezultate međutim aplikacija se mora testirati za sve vrijednosti, a za to je potreban duži vremenski period. Također, sustav je prilagodljiv i fleksibilan na moguće pogreške.

Ključne riječi: Android aplikacija, bespilotna letjelica, poplave, vodostaj

ABSTRACT

Title: A mobile application for monitoring water level of river by processing image from unmanned aerial vehicle.

Unmanned aerial vehicles like drones are becoming more popular in the world of technology just because they can "see" from perspectives that people can not. They are mostly used in commercial and military purposes, but they can also be applied in scientific purposes. This final paper shows usage of Phantom 3 Advanced drone for the purpose of measuring water level of Drava river by processing image in Android application using OpenCV library and elaborated algorithms (in this paper) based on the characteristics of river bank of Drava and own measurements to get water level. All drone features are also shown using the DJI GO application and its possible expansion and enhancement using the DJI Development KIT tool. This paper describe the whole process of capturing, saving and editing images, how does the interface of this Android application look and how are results shown in the application. Application usage is intuitive and functionally limited. The results of the analysis show promising results, but application must be tested for all water level values which requires longer period of time for testing. The application system is customizable and flexible for possible errors.

Keywords: Android application, unmanned aerial vehicle, floods, water level

ŽIVOTOPIS

Kruno Semialjac rođen je 11. travnja 1996. godine u Našicama. Osam je godina pohađao Osnovnu školu Ivane Brlić Mažuranić u Koški. 2011. godine upisuje se u Prirodoslovnu matematičku gimnaziju u Srednjoj školi Isidora Kršnjavog u Našicama. S vrlo dobrim uspjehom 2015. godine upisuje preddiplomski studij Računarstva na tada Elektrotehničkom fakultetu u Osijeku, sada Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek.

PRILOZI

Prilog 1. Završni rad u docx formatu

Prilog 2. Završni rad u pdf formatu

Prilog 3. Programski kod projekta