Optimiranje tokova energije u mikromreži s fotonaponskim sustavom temeljeno na kratkoročnom predviđanju sunčeve dozračenosti

Radovan, Aleksander

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ALEKSANDER RADOVAN

OPTIMIRANJE TOKOVA ENERGIJE U MIKROMREŽI S FOTONAPONSKIM SUSTAVOM TEMELJENO NA KRATKOROČNOM PREDVIĐANJU SUNČEVE DOZRAČENOSTI

DOKTORSKI RAD

Zagreb, 2021.

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ALEKSANDER RADOVAN

OPTIMIRANJE TOKOVA ENERGIJE U MIKROMREŽI S FOTONAPONSKIM SUSTAVOM TEMELJENO NA KRATKOROČNOM PREDVIĐANJU SUNČEVE DOZRAČENOSTI

DOKTORSKI RAD

Mentor:
Prof. dr. sc. Željko Ban

Zagreb, 2021.

University of Zagreb
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Aleksander Radovan

ELECTRICAL ENERGY FLOW OPTIMIZATION IN A MICROGRID WITH A PHOTOVOLTAIC SYSTEM BASED ON SHORT-TERM SOLAR IRRADIANCE PREDICTION

DOCTORAL THESIS

Supervisor:
Professor Željko Ban, PhD

Zagreb, 2021.

Doktorski rad izrađen je na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva, na Zavodu za automatiku i računalno inženjerstvo.
Mentor:
Prof. dr. sc. Željko Ban
Doktorski rad ima: 120 stranica
Doktorski rad br.:

O mentoru

Ban Željko je rođen 13. 09. 1962. godine u Prugovcu (općina Đurđevac, Republika Hrvatska), narodnost Hrvat, državljanstvo Hrvatsko.

Elektrotehnički fakultet Sveučilišta u Zagrebu upisao je 1981. godine, da bi 1985. diplomirao, a 1991. magistrirao na smjeru Automatika. Godine 1999. obranio je doktorsku disertaciju na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. Tijekom redovnog i postdiplomskog studija dobio je više priznanja te brončanu i srebrnu plaketu "Josip Lončar".

Nakon diplomiranja radio je godinu dana u Elektrotehničkom institutu "Rade Končar" u Zagrebu na razvoju i ispitivanju algoritama za mikroprocesorsko upravljanje istosmjernim pogonima. Od 1988. godine zaposlen je kao asistent, od 2001. godine kao docent, a od 2006. godine kao izvanredni profesor te od 2014. godine kao redoviti profesor na zavodu za Automatiku i računalno inženjerstvo Fakulteta elektrotehnike i računarstva u Zagrebu, na grupi predmeta Automatsko upravljanje sustavima. Kao asistent je sudjelovao je u izvođenju nastave na predmetima: Automatsko upravljanje, Automatsko upravljanje proizvodnim procesima, Analogna i hibridna tehnika, Elementi automatike, Industrijski roboti i fleksibilni proizvodni procesi i Modeliranje i simuliranje procesa. Od izbora za docenta do danas vodi predmete Modeliranje i simuliranje procesa te izborne predmete Optimiranje parametara sustava i Optimiranje i primjena adaptivnih regulatora te sudjeluje u nastavi predmeta Automatizacija postrojenja i procesa, Modeliranje i simuliranje sustava, Alarmni sustavi te Adaptivno i robusno upravljanje. Na Tehničkom fakultetu Sveučilišta u Rijeci vodio je predmet Modeliranje i simulacija sustava te sudjelovao u nastavi predmeta Elementi automatizacije postrojenja i Automatizacija postrojenja i procesa. Na Studiju energetske učinkovitosti i obnovljivih izvora energije Sveučilišta u Zagrebu – Šibenik organizirao je i izvodio nastavu iz predmeta Automatsko upravljanje, Sustavi za pohranu energije i Vještine – Matlab. Na postdiplomskom studiju vodio je predmet Adaptivno upravljanje primjenom referentnog modela. Pod njegovim vodstvom obranjeno je 60 diplomskih i završnih radova jedan magistarski rad i tri doktorske disertacije.

Tijekom rada na fakultetu sudjelovao je kao suradnik u znanstveno istraživačkim projektima financiranim od Ministarstva znanosti i tehnologije republike Hrvatske i to na

projektima Adaptivno i optimalno upravljanje elektromotornim pogonima te Inteligentno i adaptivno upravljanje sustavima. Nakon toga vodio je znanstvene projekte financirane od Ministarstva znanosti obrazovanja i športa Republike Hrvatske i to 2001. godine istraživački projekt pod nazivom Inteligentno i adaptivno upravljanje sustavima, a od 2006. godine projekt Sustav upravljanja energetskog izvora s kogeneracijom na osnovi gorivnih članaka. Osim toga vodio je IRI projekt „Razvoj nove generacije industrijskih modularnih, redundantnih, višeizlaznih sustava neprekidnog napajanja istosmjernim i izmjeničnim naponima“ te projekt Izgradnja prototipa punionice za električne bicikle financiran od Zagrebačkog elektrotehničkog poduzeća, a sudjelovao je i na drugim FP7 i IRI projektima te projektima financiranim od HRZZ-a na poslovima vezanim uz identifikaciju, modeliranje i simuliranje sustava automatskog upravljanja u industrijskim procesima.

Služi se engleskim jezikom.

About supervisor

Ban Željko was born on September 13, 1962 in Prugovac (Đurđevac municipality, Republic of Croatia), nationality Croat, Croatian citizenship.

He enrolled in the Faculty of Electrical Engineering at the University of Zagreb in 1981, graduating in 1985 and master's degree in Control theory in 1991. In 1999 he defended his doctoral thesis at the Faculty of Electrical Engineering and Computing, University of Zagreb. During his graduate and postgraduate studies, he received several awards and a bronze and silver plaque "Josip Loncar".

After graduation, he was employed at the Rade Končar Electrical Engineering Institute in Zagreb for a year, developing and testing algorithms for microprocessor control of DC drives. Since 1988 he has been employed as an assistant at the Department of Automation and Computer Engineering at the Faculty of Electrical Engineering and Computing in Zagreb, in the Control systems group. He was with the same institution as assistant professor, associate professor and full professor since 2001, 2006 and 2014, respectively As an assistant, he has been involved in the following courses: Automatic Control, Automatic Control of the Manufacturing Processes, Analog and Hybrid Technique, Elements of Control Systems, Industrial Robots and Flexible Manufacturing Processes, and Process Modeling and Simulation.

After obtaining the position of the assistant professor he has been lecturer of the courses Modeling and Simulation of Processes, Optimizing System Parameters and Optimizing and Applying Adaptive Controllers.

At the Faculty of Engineering, University of Rijeka, he was lecturer of the course Modeling and Simulation of Systems and took part in the lectures of course Elements of Plant Automation and Plant and Process Automation. At the Study of Energy Efficiency and Renewable Energy Sources of the University of Zagreb - Šibenik he organized and was lecturer of the courses in the fields of Automatic Control, Energy Storage Systems and Skills - Matlab. At the postgraduate level, he taught the course Reference Model Adaptive Control. Under his leadership, 60 graduation and bachelor thesis were defended, one master's thesis and three doctoral theses.

During his work at the Faculty, he participated as a researcher in scientific research projects financed by the Ministry of Science and Technology of the Republic of Croatia

on the projects Adaptive and Optimal Control of Electric Motor Drives and Intelligent and Adaptive System Control.

In addition, he led scientific projects funded by the Ministry of Science and Education of the Republic of Croatia named Intelligent and Adaptive Systems Control in 2001, and since 2006 the project Energy Management System with Cogeneration Based on Fuel Cells. In addition, he led the IRI project "Development of the next generation of industrial modular, redundant, multi-output DC and AC power systems" and the project of building a prototype of a charging station for electric bicycles, funded by the Zagreb Electrical Engineering Company. Besides, he participated as researcher in other FP7 and IRI projects and projects funded by HRZZ on tasks related to the identification, modeling and simulation of automatic control systems in industrial processes.

He is fluent in English.

Zahvala

Zahvaljujem se mentoru prof. dr. sc. Željku Banu na mentorstvu i stručnom vodstvu tijekom doktorskog studija, savjetovanja, svesrdnog prenošenja iskustva i znanja kroz pisanje doktorske disertacije.

Zahvaljujem se prof. dr. sc. Viktoru Šundeu, prof. dr. sc. Mariu Vašku i prof. dr. sc. Svenu Lončariću na brojnim savjetima tijekom pisanja doktorske disertacije.

Najveću zahvalnost želim izraziti svojoj obitelji, prije svega supruzi Martini na podršci, strpljenju i razumijevanju tijekom studiranja, istraživanja te pisanja doktorske disertacije.

Sažetak

U ovom radu je opisan postupak estimacije i predviđanja zasjenjenja Sunca analizom slijeda slika neba, koji se temelji na analizi slika snimljenim širokokutnom kamerom. Kamera je instalirana na tlu kako bi se moglo pratiti kretanje Sunca od izlaska do zalaska. Motivacija za ovo istraživanje je dizajniranje mikromreže sa solarnom energijom te različitim vrstama pohrane energije. Opisana metodologija koristi slike neba snimljene svakih pet sekundi s kojima se estimira buduće kretanje oblaka, predviđa se trenutak zasjenjenja Sunca i trajanje sjene oblaka ispred Sunca. Slike neba snimljene širokokutnom kamerom najprije su obrađene na način da se otkloni efekt distorzije širokokutne leće kamere korištenjem ugrađenog modula iz programskog paketa MATLAB, koji ne naziva Omni Directional Camera Calibration Toolbox. S obzirom na dinamičku prirodu promjene oblika, oblaci su estimirani oblikom pravokutnika kako bi sam algoritam bio što jednostavniji i brži. Obradom snimljenih slika predviđa se razina Sunčeve dozračenosti na osnovu predviđenog zasjenjenja, tj. predviđene razine vidljivog dijela Sunca te predikcije srednje vrijednosti direktne sunčeve dozračenosti od strane meteorološkog servisa. Dobiveni rezultati predviđanja sunčeve dozračenosti prikazani su u obliku krivulje količine direktne sunčeve dozračenosti u narednom periodu koji se potom koristi u postupku optimizacije tokova energije u mikromreži s fotonaponskim sustavom i sustavima pohrane zasnovanoj na kratkoročnom predviđanju Sunčeve dozračenosti i karakteristikama sustava pohrane energije. Analizirana mikromreža sastoji se i od više tipova pohrane energije s različitom učinkovitošću, kapacitetom i vremenom odziva.

Ključni pojmovi: mikromreža s fotonaponskim sustavom, obrada slike neba, otklanjanje efekta distorzije širokokutne kamere, OpenCV biblioteka, erozija i dilacija, predviđanje trenutka zasjenjenja Sunca, predviđanje Sunčeve dozračenosti, analiza slike neba, detekcija oblaka, predviđanje kretanja oblaka i Sunca, optimiranje tokova energije

Sadržaj

1. Uvod

U ovom radu opisana je metodologija za predviđanje sunčeve dozračenosti na temelju analize slike neba koje uključuju nebo, oblake i njihovo kretanje te razine prekrivanja sunca. Motivacija za ovo istraživanje je dizajniranje mikromreže sa solarnom energijom. U takvoj elektrani krivulja maksimalne granične proizvodnje za buduće razdoblje mogla bi se odrediti iz dugoročne prognoze direktne sunčeve dozračenosti po satu koju prognozira meteorološki servis. Krivulja granične proizvodnje važna je dispečeru elektroenergetskog sustava kako bi se osigurala njegova stabilnost. Prognoziranje sunčeve dozračenosti temelji se na prosječnom zračenju po satu. Kako bi se postigla planirana prosječna proizvodnja električne energije, potrebno je koristiti skladište energije za kompenzaciju odstupanja trenutne dozračenosti od prosječne vrijednosti. Optimalna upotreba uređaja za pohranu energije s različitom učinkovitošću, kapacitetom i vremenom odziva može se postići ako je dostupno preciznije predviđanje zračenja za vremenski interval sljedećeg sata. Primjer implementacije sustava za predviđanje direktne sunčeve dozračenosti korištenjem slika snimljenih komercijalnom kamerom [1], 453 puta tijekom 16 dana.

Prognoza za satnu predikciju razvijena u ovom radu koristi predviđanje kretanja oblaka analizom slika nema snimljenih širokokutnom kamerom instaliranoj na tlu koja slika nebo svakih pet sekundi. Smjer kretanja oblaka izračunava se prema kretanju njihovih centroida i detektiranih rubova oblaka unutar jedne minute (nakon analize dvanaest uzastopnih slika neba). Radi dinamičke prirode oblaka koji učestalo mijenjaju svoj oblik, koristila se aproksimacija oblika oblaka i Sunca jednostavnijim oblikom: pravokutnikom. Na temelju vidljivih oblaka na nebu u tom trenutku i njihovog kretanja, predviđa se položaj oblaka i Sunca u narednom periodu, kao i razina dozračenosti. Opisana metoda pokazala se jednostavnom i vremenski efikasnom, budući da se oblik oblaka i Sunca aproksimacijom pomoću pravokutnika temelji samo na četiri točke. Ne ovisi o sofisticiranim algoritmima i ne zahtijeva korištenje složenih modela poput onih koji se temelje na umjetnim neuronskim mrežama i na velikim podatkovnim skupovima.

Predviđanje sunčeve dozračenosti predmet je mnogih studija obavljenih proteklih godina. Neka istraživanja koristila su umjetne neuronske mreže ili duboko učenje s velikim podatkovnim skupovima. Primjeri predviđanja sunčeve dozračenosti na temelju

strojnog učenja [2]–[9] pokazali su da na predviđanje sunčeve dozračenosti može utjecati različiti broj ulaznih parametara umjetne neuronske mreže i višeslojnih perceptrona. Druga metoda strojnog učenja koja se temelji na javno dostupnim vremenskim izvještajima pokazala je pristup horizontu predviđanja od 24 sata [10]. Jedna od metoda opisala je točnost predviđanja dva dana unaprijed na temelju skupa podataka koji pokriva razdoblje od osam mjeseci [11]. Te metode omogućuju razvoj sustava koji djeluje autonomno, ali ovisi o prikupljenim podacima i treniranju modela koji se mora dodatno trenirati kako se parametri mijenjaju tijekom vremena te kako bi ispravno radio i za podatke koji su naknadno prikupljeni. Štoviše, ako je razdoblje predviđanja predugo, pristup nije prikladan za sustave mikromreža koji podržavaju različite vrste pohrane energije [12]–[15] s obzirom na potrebu dinamičke prilagodbe parametrima sustava kao što su kapacitet, trenutno opterećenje i brzine pohrana energije.

Ostale metode koriste predviđanje kretanja Sunca i oblaka upotrebom satelitskih slika i vektora kretanja kako bi se predviđala izlazna snaga, ali samo za sljedećih 30 minuta, što može biti ograničavajuće, posebno zbog toga što pohrana energije u mikromreži mogu zahtijevati više vremena za prebacivanje njihovog načina rada iz punjenja u pražnjenje [16]. Slična metoda se koristi s nekoliko kamera koje imaju leće s efektom ribljeg oka (engl. *fisheye cameras*), međutim ova metoda predviđa prekrivanja Sunca između 15 i 30 minuta s ažuriranjem situacije svake minute [17]–[19]. Metoda razvijena u ovom radu temelji se na predviđanju razine solarne dozračenosti do sat vremena.

Razvoj algoritma detekcije oblaka pomoću slika neba je također korišten u raznim varijantama. Jedan od pristupa razvoja algoritma koristi reduciranje interferencije sunčeve svjetlosti u slikama snimljenim kamerom koja se nalazi na tlu (engl. *ground-based sky images*) [20]. Također postoje istraživanja korištenjem slike neba s oblacima koja se uspoređuje sa slikom neba bez oblaka [21]. Detekciju oblaka je moguće provesti i s različitim vremenima ekspozicije te korištenjem adaptivnog praga (engl. *threshold*) [22]. Korištenjem HSV (*Hue, Saturation, Value*) prostor boja daje najbolje rezultate u detekciji oblaka i Sunca, njihovih kontura i centroida na temelju čega se određuje vektor njihovog gibanja. Za razliku od Canny detektora rubova oblika na slici koji zahtijeva nekoliko sekundi za obrađivanje pojedine slike, algoritam s korištenjem HSV prostora zahtijeva nekoliko desetaka milisekundi za obradu slike i prikladan je za brzinu predviđanja koja je potrebna za algoritam opisan u ovom radu, između pet i šest sekundi

za cijelu sliku. Optimalne razine HSV veličina u svrhu automatiziranja obrade slike i detekciju kontura Sunca i oblaka mogu biti postavljene od strane umjetne neuronske mreže koja je korištena u radu [23]. Iz slika neba mogu se izdvojiti različite značajke temeljem kojih se definiraju parametri umjetne neuronske mreže kao što je broj oblaka, veličina oblaka, vrijednosti histograma koji sadrži RGB (*Red, Green, Blue*) vrijednosti i slično, te boja koje dominiraju na slikama neba i prema kojima se definiraju ostali parametri poput strukturnih elemenata na morfološke operacije erozije (engl. *erosion*) i dilacije (engl. *dilation*). Ostale varijante detekcije oblaka na slikama neba koje sadrže Sunce i oblake snimljene kamerom postavljene na tlu koriste segmentaciju oblaka [24], koja koristi treniranje modela i nije najprikladnija za slučajeve naglih promjena vremenskih prilika. Postoje istraživanja koja uključuju umjetne neuronske mreže ovisne o podatkovnim skupovima (engl. *datasets*) koji su prikupljeni na određenom geografskom području [25],[26], te nisu prikladna za korištenje na svim područjima, već samo na onima gdje su dostupni ti podatkovni skupovi.

Rješenje temeljeno na IoT (engl. *Internet of Things*) platformi predviđa kretanja oblaka unutar jedne minute pomoću pomaka njihovog centroida te tu informaciju koristi za upravljanje izlaznom energijom sustava sastavljenog od fotonaponskih modula [27]. Predloženo rješenje se fokusira na kratkoročnu predikciju kretanja oblaka i trenutka prekrivanja Sunca. Kratkoročna predikcija u ovom slučaju znači predviđanje kraće od sat vremena, za razliku od sustava opisanog u ovom radu koji previđa prekrivanje Sunca u sljedećih sat vremena. Sustav opisan u ovom radu se sastoji od pohrana energije unutar mikromreže (engl. *microgrid*) koja se razlikuju po svojstvima kao što su brzina odziva, učinkovitost i kapacitet. Period od sat vremena je bitan zbog toga što se referencira na predviđanje srednje vrijednosti satne dozračenosti od strane hidrometeorološkog zavoda u mjernoj jedinici $\frac{\mathrm{W}}{\mathrm{m}^2}$.

Satelitske slike zajedno s podrškom vektorske sheme temeljenje na strojnom učenju također mogu biti korištene u predviđanju količine dozračene energije što je opisano u radu [28], međutim ovaj pristup nije adekvatan za solarne sustave koji su locirani na određenoj geografskoj lokaciji.

Za predviđanje kretanja oblaka mogu se koristiti i algoritmi s prepoznavanjem blokova piksela u određenoj regiji u sljedećoj slici te detekcijom tih istih blokova koji imaju najviši stupanj sličnosti [29], što potvrđuje da je analiziranje slike neba pouzdan

način za predikciju kretanja oblaka. Jedini izazov kod takvog pristupa je duže vrijeme procesiranja slika, 20 sekundi, zbog čega se ovakav pristup ne može koristiti u stvarnom vremenu. Za razliku od njega, pristup opisan u ovom radu je prilagođen za korištenje u stvarnom vremenu, jer se predikcija kretanja oblaka u sljedećih sat vremena određuje u samo nekoliko sekundi.

Pretpostavka korištena u ovom radu je da je dugoročno predviđanje sunčeve dozračenosti sa satnom rezolucijom, dano od meteorološkog servisa, dovoljno točno te da se može koristiti kao referenca za gornju granicu planirane isporuke energije koja mora biti usklađena s distributerom električne energije. Kako bi se mogla pohraniti deklarirana razina proizvedene energije unutar satnog intervala, koja je jednaka srednjoj vrijednosti energije predviđene od meteorološkog servisa, potrebno je koristiti sustav pohrane energije s različitim vrstama pohrana energije. Ove pohrane se razlikuju po kapacitetu, učinkovitosti i brzini odziva. Efikasno korištenje pohrana energije je moguće ako je dostupno precizno predviđanje sunčeve dozračenosti u maksimalnoj razlučivosti od jedne minute i u sljedećih sat vremena. S obzirom na to da kamera slika nebo svakih pet sekundi, a predviđanje kretanja oblaka i Sunca koristi sekvence od dvanaest slika, definirani period analize, tj. razlučivost iznosi jednu minutu (5 sekundi x 12 = 60 sekundi = 1 minuta). Kako bi se to ostvarilo, potrebno je odrediti krivulju koja prikazuje postotak prekrivenosti Sunca oblacima, tj. vidljivosti Sunca kroz vrijeme unutar sat vremena. Normaliziranje srednje vrijednosti te krivulje s krivuljom srednje dozračenosti koju predviđa hidrometeoroški zavod za vrijeme sata predviđanja daje preciznu krivulju dozračenosti kojom je moguće upravljati sustavom s različitim pohranama energije. Krivulja predviđene dozračenosti koja se optimizirana korištenjem koeficijenata koji opisuju karakteristike energetskog proizvodnog pogona može poslužiti kao temelj za on-line optimizaciju korištenja energetskog sustava.

U drugom poglavlju opisan je koncept mikromreže temeljene na solarnoj energiji s različitim vrstama pohrane.

U trećem poglavlju opisan je način obrade slike koji uključuje ispravljanje distorzije slike neba snimljenih širokokutnom kamerom. Osim toga su opisane tehnologije i alati pomoću kojih su razvijeni algoritmi, od programskog paketa MATLAB do programskog jezika Java te biblioteke za obradu slike i računalni vid, OpenCV. Također su prikazane

i značajke modela boja koji se koristio za detekciju Sunca i oblaka na nebu, kao i dijagrami toka koji oslikavaju sve korake algoritma koji su korišteni kod obrade slike.

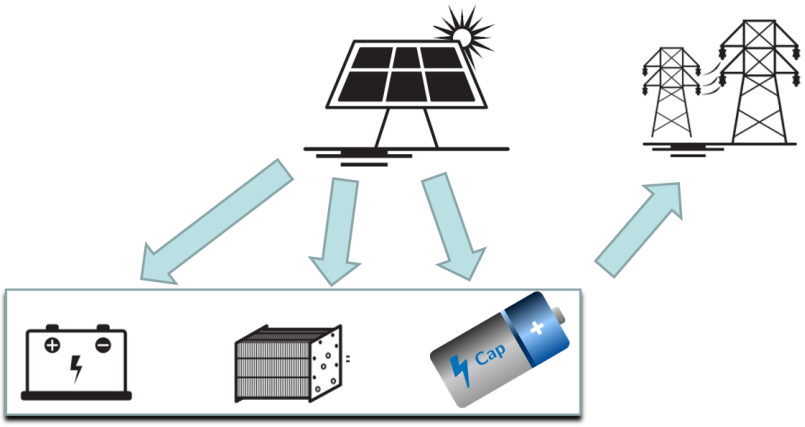
Četvrto poglavlje opisuje algoritam izračuna razine zasjenjenja Sunca te način na koji su se oblici sunca i oblaka aproksimirali oblikom pravokutnika kako bi početak, trajanje, razina i završetak zasjenjenosti mogli biti određeni što točnije.

Zasjenjenost utječe na razinu direktne dozračenosti Sunca, a način određivanje dozračenosti je opisan u petom poglavlju. Od svih oblaka na nebu detektiraju se samo oni oblaci koji svojom putanjom mogu prekriti površinu Sunca pa je na početku potrebno odrediti kandidate koji mogu zasjeniti Sunce s obzirom na vektor brzine i smjer njihovog gibanja. Nakon toga, upotrebom tog vektora se simulira gibanje oblaka i Sunca u sljedećem periodu (dok se još uvijek nalaze na obzoru koji obuhvaća kamera, tj. slika neba bez efekta distorzije). Za svaku novu poziciju oblaka se određuje nova razina zasjenjenosti Sunca. Na kraju petog poglavlja prikazani su rezultati koji prikazuju točnost predikcije direktne dozračenosti na temelju koje je moguće upravljati načinom pohranjivanja energije u mikromreži.

Šesto poglavlje se fokusira na dimenzioniranje i optimiranje sustava sastavljenog od tri različite pohrane energije: superkondenzatora, baterije i vodika. Prikazana je ovisnost iznosa krivulje energije o različitim oblicima krivulje snage. Osim toga su objašnjene razlike u karakteristikama između različitih tipova pohrane energije te je prikazan MATLAB/Simulink model za optimiranje tokova energije. Na kraju su prikazani rezultati koji opisuju uspješnost upravljanja tokovima energije.

2. Mikromreže s različitim pohranama energije

U ovom poglavlju opisana je mikromreža koja koristi fotonaponske module za proizvodnju električne energije iz sunčeve svjetlosti i različite pohrane poput baterija, superkondenzatora i/ili vodika. Svaki od ovih pohrana energije se razlikuje po parametrima kao što su kapacitet, brzina odziva i učinkovitosti. S obzirom na ove parametre definira se način upravljanja samim energetskim sustavom. Primjer takve mikromreže prikazan je na slici 2.1.



Slika 2.1: Mikromreža s različitim vrstama pohrane energije

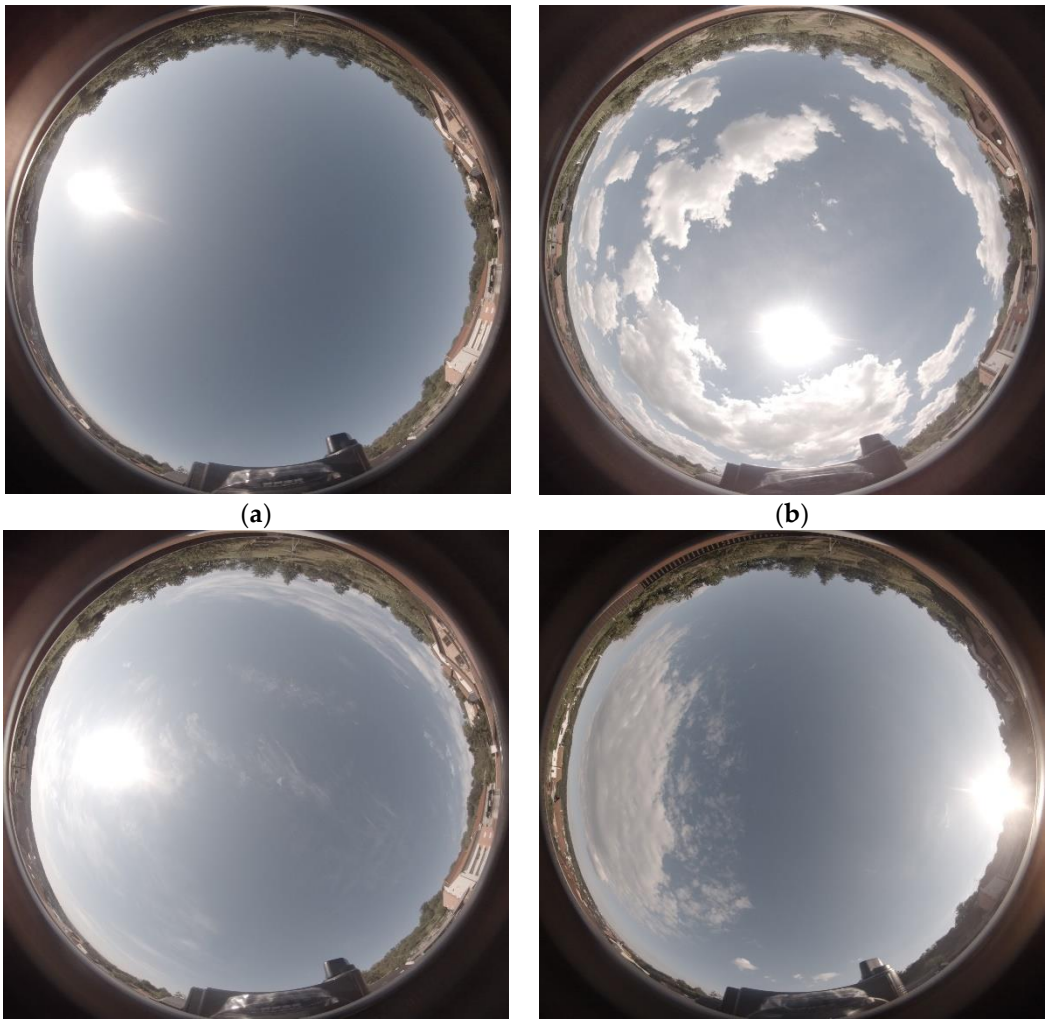
Za dispečera elektroenergetskog sustava važno je osigurati stabilnost elektroenergetskog sustava. Da bi se postigla prosječna vrijednost proizvodnje energije, potrebno je koristiti adekvatna skladišta energije kako bi se nadoknadilo odstupanje trenutne dozračenosti od prosječne vrijednosti te time očuvala stabilnost elektroenergetskog sustava. Optimalno korištenje skladišta energije može se postići u slučaju dostupnosti preciznijeg predviđanja sunčeve dozračenosti tijekom vremenskog intervala od sljedećeg sata.

Optimalna upotreba različitih sustava za pohranu energije zahtijeva precizno predviđanje sunčeve dozračenosti s barem satnim vremenskim horizontom. Da bi se to postiglo, algoritam predviđanja kretanja oblaka mora se koristiti za generiranje krivulje koja pokazuje razinu vidljivosti Sunca na nebu u sljedećem razdoblju i na temelju toga, predvidjeti sunčevu dozračenost u sljedećih sat vremena. Korištenjem dugoročne prognoze zračenja po satu od meteorološkog servisa može se odrediti direktna sunčeva dozračenost. Generirana krivulja koja pokazuje razinu vidljivosti Sunca u sljedećem periodu, na primjer 60 minuta u ovom radu, iskorištava se kako bi se na temelju predviđene srednje dozračenosti na razini sata predviđene od strane meteorološkog

servisa odredila dinamika promjene dozračenosti unutar 60 minuta. Ta dinamika je predstavljena je krivuljom koja na osi apscise ima vrijeme, a na osi ordinate ima mjeru dozračenosti predstavljenoj u jedinici $\frac{\mathrm{W}}{\mathrm{m}^2}$.

3. Obrada slike neba

Analiza slike neba korištena u ovom radu u svrhu praćenja kretanja oblaka i Sunca temelji se na slikanju neba u razmaku od pet sekundi. Takva frekvencija je odabrana na osnovu dinamike kretanja oblaka i Sunca, te se značajnije promjene pozicije oblaka mogu zamijetiti tek nakon nekoliko sekundi. Primjeri snimljenih slika prikazani su na slici 3.1.



Slika 3.1: Primjeri slika neba: (a) slika čistog neba samo sa Suncem i bez oblaka snimljeno 13.07.2020. u 08:09:09, (b) slika sa Suncem i kumulus oblacima snimljena 13.07.2020. u 13:26:20, (c) slika sa Suncem i cirus oblacima snimljena 14.07.2020. u 09:09:15 te (d) slika snimljena krajem dana prije zalaska Sunca snimljena 14.07.2020. u 18:45:32.

Kod promjene pozicije Sunca je taj period još veći i značajne promjene lokacije Sunca je moguće primijetiti tek nakon 10-tak minuta. Tijekom istraživanja u ovom radu

koristio se i pristup snimanja videa stanja na nebu te ekstrahiranja slika iz njega svakih nekoliko sekundi, međutim taj pristup nije dao adekvatne rezultate, prvenstveno po pitanju brzine samog algoritma.

S obzirom na korištenje širokokutne kamere, prvi korak u obradi slike je otklanjanje efekta distorzije slike, tj. određivanje parametara algoritma za otklanjanje distorzije kroz postupak kalibracije kamere. Za tu namjenu je korištena šahovska ploča dimenzija 100 cm x 90 cm koja sadrži kvadratiće veličine 2 cm x 2 cm što je prikazano u sljedećem poglavlju.

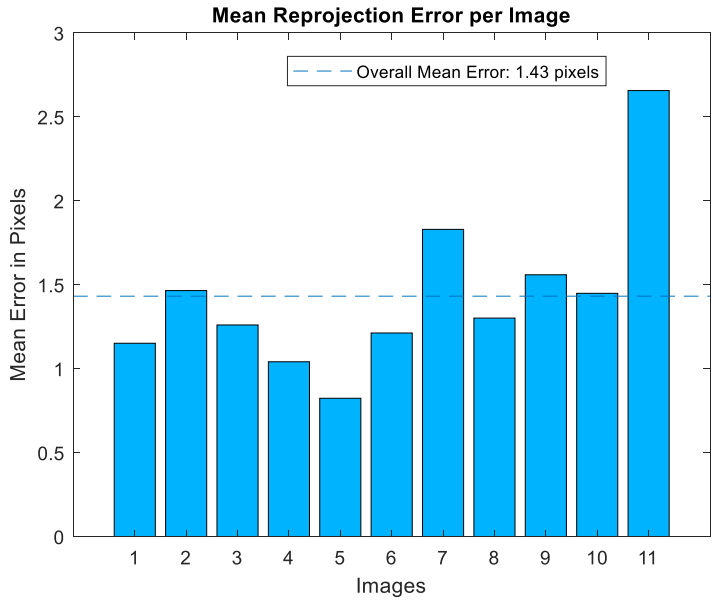
3.1. Ispravljanje efekta distorzije slike

Za ispravljanje efekta distorzije slike korištenjem širokokutne kamere GoPro Fusion korišten je MATLAB-ov alat Omni Directional Camera Calibration Toolbox [30]–[32]. Ovaj alat omogućuje korisniku kalibriranje bilo koje širokokutne kamere u dva koraka: (i) priprema i obrada slike šahovske ploče prikazane u različitim položajima i orijentacijama te (ii) detektiranje vrhova kvadratića na šahovskoj ploči. Upotrebom detektiranih podataka određuju se svi ostali parametri potrebni za ispravljanje efekta distorzije slike i kreiranje neiskrivljene slike.

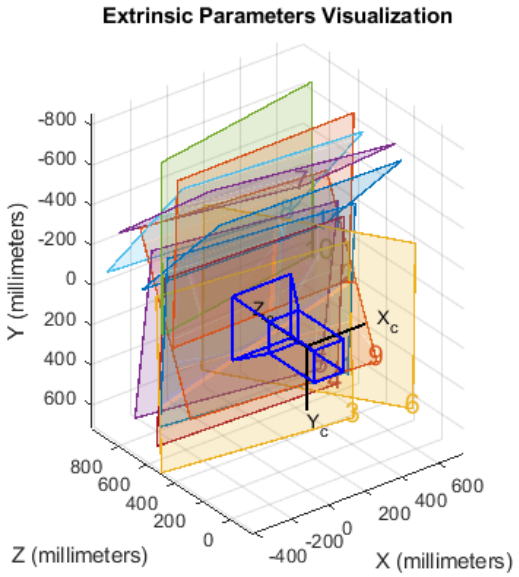
MATLAB-ov alat Omni Directional Camera Calibration Toolbox, nakon kalibracije, omogućava funkcije za definiranje odnosa između zadane točke piksela i njegove projekcije na jedinstvenu sferu. Postupak kalibracije započinje odabirom slika s uzorcima šahovskih ploča i nastavlja se otkrivanjem točaka šahovskih ploča pomoću metode "detectCheckerboardPoints". Veličina slike određuje se na temelju prve pročitane slike. Razlučivost slika korištenih u ovom radu bila je 3104 x 3000 piksela. Korištene slike šahovske ploče prikazane su na slici 3.4.

U sljedećem koraku određuju se koordinate kutova kvadrata u stvarnom prostoru metodom "generateCheckboardPoints". Nakon toga obavlja se kalibracija upotrebom parametara ribljeg oka (engl. *fisheye*) metodom "estimateFisheyeParameters". Posljednja faza je generiranje neiskrivljene slike metodom "undistortFisheyeImage". Osim navedenih metoda još je moguće dohvatiti informacije o pogreškama kod preslikavanja

točaka slike (engl. *reprojection errors*) pomoću metode „showReprojectionErrors“ što je prikazano na slici 3.2. S obzirom na to da je za kalibraciju kamere korišteno 11 slika (koje su prikazane na x osi, „Images“), za svaku od tih slika prikazana je razina pogreške te usporedba sa srednjom pogreškom (što je prikazano u pikselima, „MeanErrorsinPixels“). Također je pomoću metode „showExtrinsics“ moguće prikazati pozicije šahovskih ploča koje su korištene kod detekcije parametara distorzije korištene kamere, što je prikazano na slici 3.3.



Slika 3.2: Srednja pogreška kod određivanja neiskrivljene slike generirana pomoću MATLAB funkcije „showReprojectionErrors“



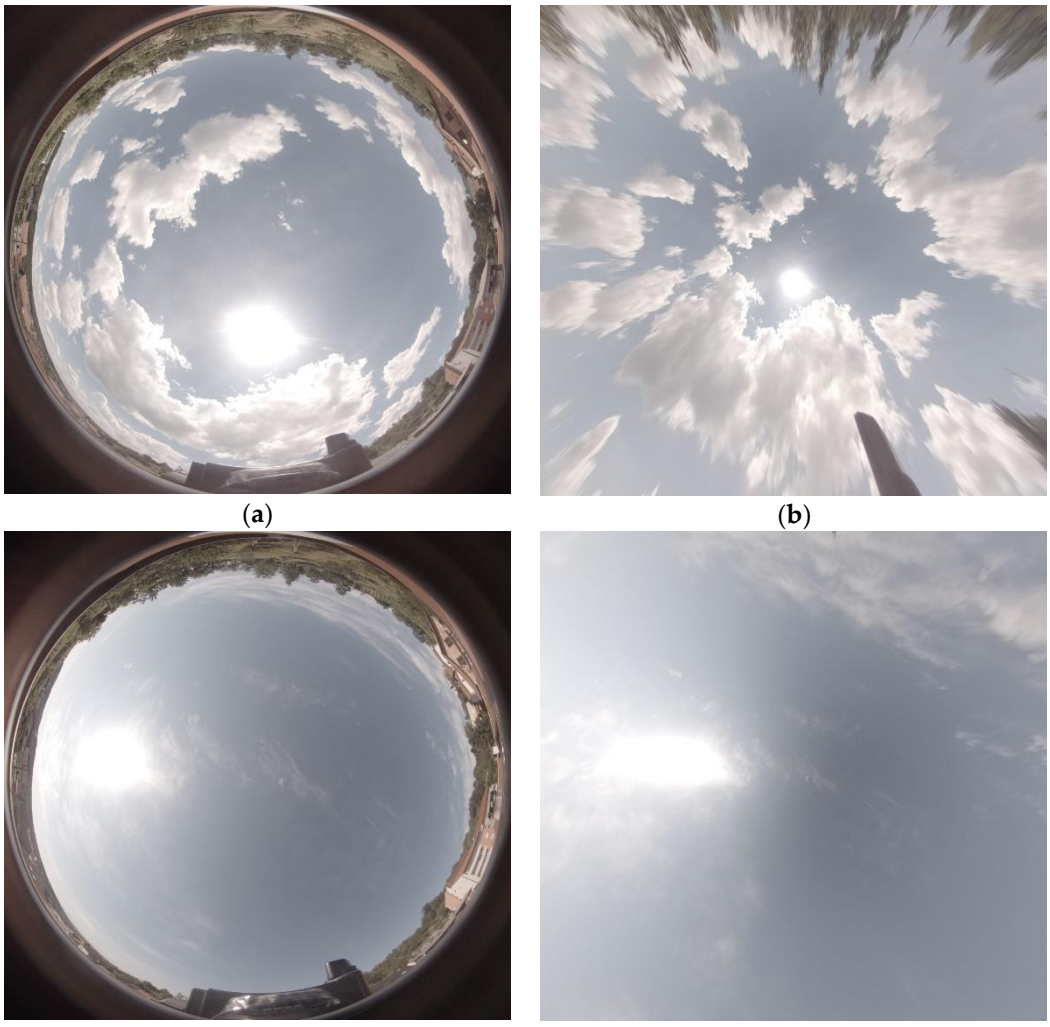
Slika 3.3: Prikaz detektiranih položaja šahovskog polja korištenjem MATLAB funkcije „showExtrinsics“

(**a**) (**b**)

Slika 3.4: Prikaz rezultata odstranjivanja efekta distorzije slike širokokutnom kamerom GoProFusion: (a) iskrivljena slika, (b) neiskrivljena slika

Na slici 3.4 (a) prikazana je iskrivljena šahovnica koja se koristi za kalibraciju kamere, a na slici 3.4 (b) prikazana je neiskrivljena verzija nakon provedenog opisanog postupka. MATLAB skripta koja sadrži sve naredbe za izvršavanje procesa otklanjanja prikazana je u Prilogu 1 na kraju rada. Sve slike šahovske ploče korištene za kalibriranje kamere prikazane su u Prilogu 2, uz prikaz detektiranih kvadratića šahovske ploče koji se dobiva korištenjem CameraCalibrator aplikacije unutar MATLAB okruženja.

Na slici 3.5. prikazane su iskrivljene i pripadajuće neiskrivljene slike dobivene MATLAB skriptom prikazanom u Prilogu 1.



(**a**) (**b**)

(**c**) (**d**)

Slika 3.5: Prikaz iskrivljene i neiskrivljene slike neba: (a) iskrivljena slika u slučaju kad je Sunce na sredini slike, (b) neiskrivljena slika u slučaju kad je Sunce na sredini slike, (c) iskrivljena slika u slučaju kad je Sunce na lijevoj (istočnoj) strani slike i (d) neiskrivljena slika u slučaju kad je Sunce na lijevoj (istočnoj strani slike)

3.2. OpenCV biblioteka

OpenCV je biblioteka koja sadrži veliki broj algoritama za obradu slike i implementaciju značajki računalnog vida u realnom vremenu i napisana je u programskom jeziku Python. Radi se o biblioteci otvorenog koda (engl. *open source*) koja je dizajnirana za korištenje u aplikacijama u kojima je rad u realnom vremenu od presudne važnosti [33].

S obzirom da je algoritam za obradu slike implementiran u programskom jeziku Java, korištena je inačica OpenCV biblioteke koja je također implementirana u programskom jeziku Java, a naziva se JavaCV. U ovom radu su korištene funkcionalnosti konverzije slike iz RGB (*Red, Green, Blue*) formata u HSV (*Hue, Saturation, Value*). Obavljene su morfološke operacije erozije (engl. *erosion*) i dilacije (engl. *dilation*), detekcija kontura, računanje površine kontura (u našem slučaju oblaka i Sunca). Obavljeno je određivanje centroid točaka kontura, određivanje pravokutnika minimalne površine koji omeđuje konturu čime se aproksimira nepravilan oblik oblaka i Sunca te pojednostavljuje složenost algoritma. Također je obavljeno i određivanje površine presijecanja aproksimiranih pravokutnika. Za definiranje samog algoritma korišten je programski jezik Java.

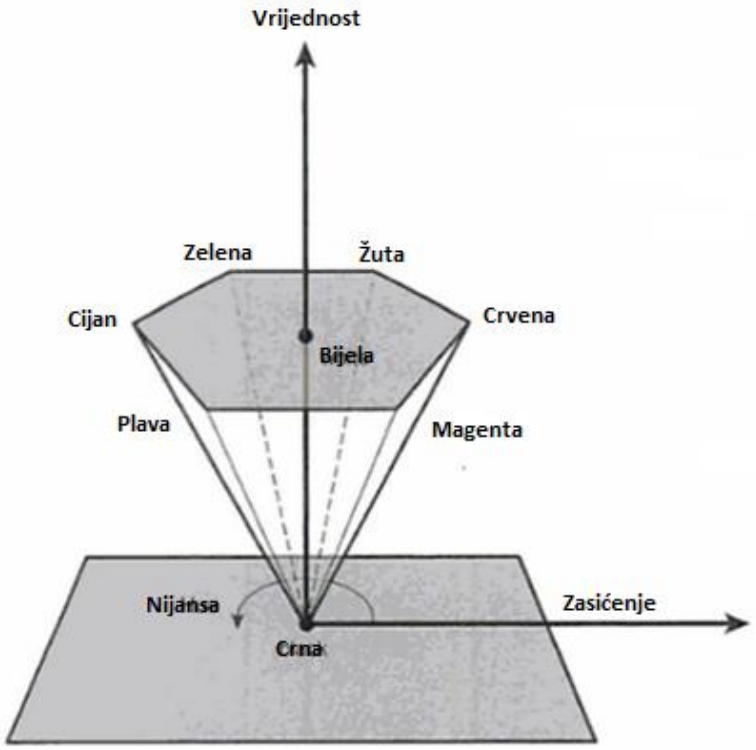
3.3. Programski jezik Java

Programski jezik Java spada u skupinu objektno orijentiranih jezika pomoću kojeg se razvijaju stolne (engl. *desktop*), mrežne (engl. *web*) i mobilne (engl. *mobile*) aplikacije. Razvoj tog jezika započeo je početkom 90-tih godina i trenutačno je jedan od najpopularnijih programskih jezika [34]. Izvršava se u Java virtualnom stroju (engl. *Java Virtual Machine*) koji mu osigurava prenosivost na različite operacijske sustave poput Microsoft Windows i Unix. Temelj prenosivosti je međukod pod nazivom bajtkod (engl. *bytecode*) koji nastaje prevođenjem Java koda. Bajtkod nije izvršni kod već visokooptimiziran skup instrukcija predviđen za izvođenje pomoću Java virtualnog stroja. U ovom radu programskim jezikom Java implementiran je algoritam za obradu slika neba koji se temelji na JavaCV biblioteci. Prvi dio programa bavi se obradom slika, a drugi dio programa generira CSV datoteku prema kojoj se generira krivulja prekrivanja Sunca oblacima.

3.4. HSV model boja

Za razliku od uobičajenog RGB modela boja, koji se temelji na stvaranju različitih nijansi boja dodavanjem različitih intenziteta crvene, zelene i plave (u rasponu od 0 do 255), u

ovom radu koristi se model HSV. HSV model boje koristi tri parametra: nijansa (engl. *Hue*), zasićenost (engl. *Saturation*) i vrijednost (engl. *Value*). Nijansa definira boju, zasićenost definira količinu boje, a vrijednost definira razinu svjetline boje ili količinu crne boje dodane u boju. Umjesto kartezijevog koordinatnog sustava koji se koristi u slučaju RGB modela boja, HSV koristi cilindrični koordinatni sustav, slika 3.5, [35].
Nijansa određuje kutove crvene boje koja je na 0˚, žute boje koja je na 60˚, zelene boje koja je na 120˚, cijan boje koja je na 180˚, plave boje koja je na 240˚ i magente koja je na 300˚. Zasićenje s vrijednošću 1,0 rezultira maksimalnom količinom boje, a niže vrijednosti rezultiraju bojom koja je bliža bijeloj boji. U slučaju vrijednosti zasićenja od 0,0, javlja se sjena sive boje, a to ovisi o razini vrijednosti parametara HSV modela boje. Na primjer, ako je boja postavljena na crvenu, zasićenje i vrijednost postavljene na 1,0, rezultirajuća boja je čisto crvena. U slučaju smanjenja zasićenja crvena boja postaje svjetlija i na kraju postaje ružičasta, a s dodatnim smanjenjem zasićenja postaje bijela.



Slika 3.6. Cilindrični koordinatni sustav HSV modela boja, izvor [35]

Pretvorba vrijednosti RGB komponente u model HSV boje započinje dodjeljivanjem maksimalne vrijednosti RGB komponente parametru vrijednosti HSV modela boje. Vrijednost parametra zasićenja određuje se izračunavanjem razlike između maksimalne i

minimalne vrijednosti RGB komponenata, a ta se razlika mora podijeliti s najvećom vrijednošću među RGB komponentama. Prvi posebni slučajevi predstavljaju vrijednosti RGB postavljene na 0, što predstavlja crnu boju, a zasićenost i vrijednost HSV parametara također imaju vrijednost 0 (u ovom slučaju vrijednost parametra nijansa nema utjecaja na rezultirajuću boju). Drugi poseban slučaj predstavlja sve RGB vrijednosti kao jednake, što predstavlja nijansu sive boje, što znači da je vrijednost zasićenja 0. Parametar nijanse u HSV modelu boja određuje se jednom od metoda prikazanih u sljedećim pojmovima, ovisno o vrijednostima RGB komponenata. Kompletni algoritam za određivanje vrijednosti HSV prikazan je u nastavku (3.1) [35]:

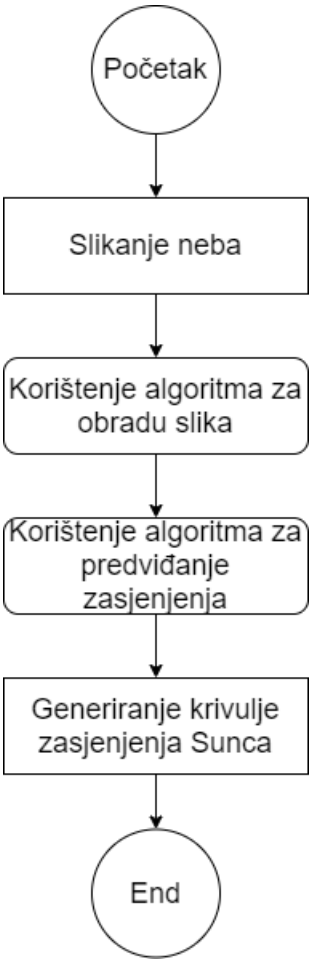
$$
\begin{aligned}
&max\_RGB = max(R, G, B)\\
&min\_RGB = min(R, G, B)\\
&d = max\_RGB - min\_RGB\\
&V = max\_RGB\\
&S = \begin{cases} 0, d = 0 \\ \dfrac{d}{max\_RGB}, d \neq 0 \end{cases}\\
&H = \begin{cases} nedefiniran, S = 0 \\ 60° \times \left(\dfrac{G-B}{d} mod\ 6\right), R = max\_RGB \\ 60° \times \left(\dfrac{B-R}{d} + 2\right), G = max\_RGB \\ 60° \times \left(\dfrac{R-G}{d} + 4\right), B = max\_RGB \end{cases}
\end{aligned}
\tag{3.1}
$$

gdje je max_RGB najveća vrijednost od jedne pojedinačnih komponenti između R, G i B koja je ujedno dodijeljena i vrijednosti V, min_RGB najmanja vrijednost od jedne od pojedinačnih komponenti između R, G i B, d predstavlja razliku između vrijednosti max_RGB i min_RGB, a preostale komponente H i S parametara poprimaju definirane prikazanim izrazima u slučajevima kad je jedna od komponenti R, G i B postavljena na vrijednost max_RGB.

Prednost HSV modela boja je bolja prezentacija boja ljudskoj percepciji, nego što se to postiže RGB modelom. U ovom radu je taj model prvenstveno odabran zbog toga što se koristi i za odvajanje osvjetljenja slike od samih boja, kako bi se što efektivnije detektirale nijanse oblaka i Sunca na slikama neba.

3.5. Algoritam obrade slike

Slika 3.7 prikazuje glavni algoritam korišten za obradu slike, koji uključuje prikupljanje slika neba, obradu slika neba, simulaciju kretanja oblaka i predviđanje trenutka prekrivanja Sunca od strane oblaka kao i trajanja sjene te generiranje krivulje koja pokazuje stupanj zasjenjenja Sunca u sljedećem razdoblju. Sadrži dva podalgoritma, od kojih se jedan usredotočuje na obradu slike, što će biti objašnjeno u ovom poglavlju, a drugi na procjenu pokrivenosti Sunca, što je objašnjeno u poglavlju 4.1.



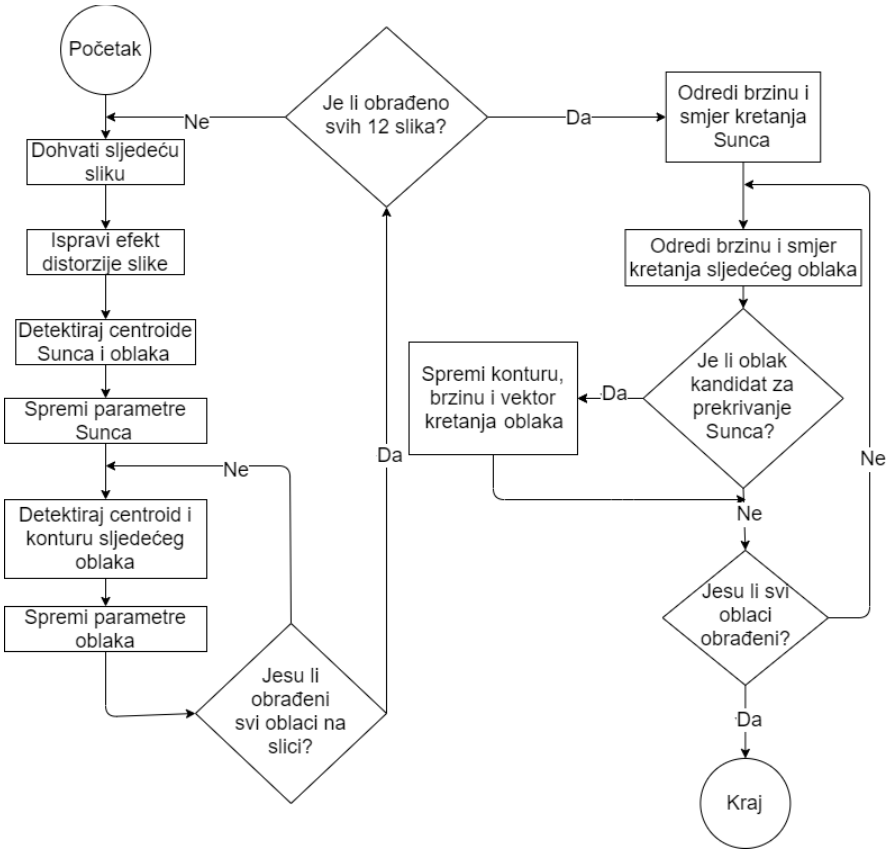
Slika 3.7 Glavni algoritam za predviđanje zasjenjenja Sunca

Metodologija korištena za otkrivanje kontura i centroida opisana je u poglavljima 3.7 i 3.8 respektivno. Kako bi se algoritam dodatno pojednostavnio i maksimalno ubrzao, konture oblaka i Sunca aproksimiraju se pravokutnim oblicima. Oblik pravokutnika svodi se na četiri točke koje predstavljaju vrhove pravokutnika. Takav pojednostavljeni oblik je moguće lakše translatirati na slici i simulirati njegovo kretanje u budućem razdoblju,

što štedi količinu procesorskog vremena potrebnog za obradu slika i predikciju dozračenosti. Pomaci točaka koji predstavljaju vrhove pravokutnika se simuliraju jednostavnim dodavanjem iznosa pomaka koordinatama x i y točaka.

Da bi se algoritam dodatno optimizirao, uzimaju se kao kandidati u obzir samo oblaci koji teoretski mogu pokriti Sunce u budućem razdoblju. Odabir oblaka temelji se na njihovom položaju (zanemaruju se svi oblaci čiji položaj i smjer kretanja ne mogu rezultirati prekrivanjem Sunca). Kandidati za oblak koji mogu prekriti Sunce pohranjuju se na daljnju obradu.

Dijagram toka algoritma za obradu slike spomenut u prethodnom dijelu poglavlja prikazan je na slici 3.8.



Slika 3.8. Algoritam za obradu slike

Prva faza korištenog algoritma je otklanjanje distorzije slike, a zatim detektiranje konture Sunca, centroida sunčeve konture, a zatim konture svih oblaka na slici i njihovih odgovarajućih centroida. Kako bi se unaprijedili rezultati detekcija Sunca i oblaka, u obzir se uzimaju samo detektirani oblici na slici sa značajnijim iznosima površina u pikselima. Za konturu Sunca kao prag za detekciju je uzeta donja granica od 5000 piksela što se

smatra iznosom površine neprekrivenog Sunca. Kad oblak počne djelomično prekrivati Sunce, površina Sunca se smanjuje, a kada je iznos površine manji od spomenutih 5000 piksela, smatra se da je Sunce prekriveno oblacima. Prag se temelji na rezoluciji kamere (3104 x 3000 piksela) i vidljivom sunčanom području na vedrom ili nepokrivenom nebu. Određuje se izračunatom površinom Sunca tijekom razdoblja snimanja neba i korištenjem minimalne vrijednosti površine kada je Sunce vidljivo i nije prekriveno oblacima.

Kao i kod praga površine Sunca postavljene na 5000 piksela, kod detekcije oblaka uzimaju se u obzir samo oblaci sa značajnom površinom, jer u slučaju premale veličine, tj. površine oblaka, ne dolazi do značajnijeg prekrivanja Sunca. Prag površine oblaka postavljen je na 3000 piksela. Oblaci s površinom manjom od 3000 piksela jednostavno se zanemaruju i ne pohranjuju u ArrayList zbirku u programskom jeziku Java s ostalim oblacima koji mogu značajnije prekriti Sunce. Prag područja oblaka temelji se na pragu područja Sunca: ako izračunato područje oblaka iznosi 60% izračunatog područja Sunca, taj se slučaj smatra značajnom razinom pokrivenosti.

Simulacija kretanja oblaka i Sunca se temelji na tendenciji kretanja unutar jedne minute, tj. obrađenih 12 slika, što predstavlja kretanje oblaka i Sunca unutar jedne minute (12 x 5 sekundi = 60 sekundi, s obzirom da se slikanje neba obavlja svakih pet sekundi). Upotrebom informacija o obliku oblaka, brzine i vektora smjera kretanja, određuje se pozicija svakog od oblaka s vremenskim odmakom od pet, deset, petnaest, dvadeset minuta i tako dalje, sve dok je oblak još uvijek prisutan u vidljivom obzoru kamere. S obzirom na poziciju Sunca, oblaci koji nisu kandidati da će prekrivati Sunce (već su prošli područje gdje se Sunca nalazi ili je vektor smjera i oblaka usmjeren u smjeru koji zaobilazi sunce), se ne razmatraju i neće se obrađivati u nastavku algoritma, što dodatno optimizira vrijeme trajanja algoritma.

3.6. Morfološke operacije erozije i dilacije

Nakon pretvorbe iz RGB u HSV model boja opisan u poglavlju 3.4. potrebno je obaviti morfološke operacije erozije i dilacije kako bi se precizno definirale granice kontura oblaka i Sunca na slici neba. Morfološke operacije koriste strukturne elemente zadane veličine i oblika na temelju vrijednosti susjednog piksela i izvode transformacije na ulaznoj slici.

Prvi korak je detektiranje konture Sunca na nebu izvođenjem morfoloških operacija erozije. Da bi se detektirala pozicija Sunca, najprije se trebaju obaviti sljedeće radnje obrade slike (kao što je opisano u našem prethodnom radu [36]): (i) pretvaranje slike iz RGB (crvena - *Red*, zelena - *Green* i plava - *Blue*) u HSV model boje (nijansa - *Hue*, zasićenost - *Saturation* i vrijednost - *Value*), (ii) postavljanje veličine strukturnih elemenata za eroziju, (iii) pronalaženje kontura na slici koje predstavljaju Sunce i (iv) pronalazak centroida Sunca.

Morfološke operacije na binarnoj slici stvaraju novu binarnu sliku u kojoj pikseli imaju vrijednost različitu od nule samo ako element strukturiranja može stati između susjednih piksela koji se provjeravaju ili se siječe sa susjednim pikselima. Element strukturiranja je binarna slika predstavljena matricom piksela koji imaju vrijednosti nula ili jedan. Dimenzije matrice ukazuju na veličinu elementa strukturiranja. Erozija binarne slike opisana je sljedećom jednadžbom:

$$n = i \ominus s \tag{3.2}$$

gdje n predstavlja novu binarnu sliku nakon operacije erozije, i početna binarna slika koja je transformirana, a s predstavlja element strukturiranja.

Na rezultirajućoj slici su jedinice na koordinatama (x, y) ishodišta elementa za strukturiranje s gdje se podudara s ulaznom slikom i, odnosno element za strukturiranje se ne nalazi na rubovima objekata na slikama, što rezultira s $n(x,y)=1$. Ako se element za strukturiranje nalazi na rubovima objekata, onda to rezultira s $n(x,y)=0$. U slučaju detektiranja Sunca, erozija uklanja oblake s binarne slike, a u slučaju detektiranja oblaka uklanja vrlo male oblake i smanjuje veličinu oblaka kako bi naglasila granice između oblaka. Operacija dilacije ima suprotan učinak od erozije: dodaje sloj piksela i na unutarnju i vanjsku granicu kontura oblaka.

3.6.1. Erozija

Prema [37], morfološka operacija erozije se može opisati kao transformacija koja se temelji na oduzimanju vrijednosti vektora. Ako su A i B skupovi u Euklidskom N-prostoru (E^N), tada erozija skupa A skupom B predstavlja skup svih elemenata x za koji

vrijedi $x + b \in A$ za svaki $b \in B$. Erozija skupa *A* skupom *B* označava se $A \ominus B$ i definiran je na sljedeći način:

$$A \ominus B = \{x \in E^N | x + b \in A \text{ za svaki } b \in B\} \quad (3.3)$$

Ako se u izraz (3.3) uvrsti izraz za razliku elemenata *a* i *b*, onda se može pisati i na sljedeći način:

$$A \ominus B = \{x \in E^N | \text{ za svaki } b \in B, \text{ tada postoji } a \in A \text{ takav da je } x = a - b\} \quad (3.4)$$

Transformacija erozije se može definirati na sljedeći način: erozija slike *A* sa strukturnim elementom *B* je set svih elemenata *x* od E^N za koji vrijedi da je skup *B* translatiran za *x* sadržan u skupu *A*:

$$A \ominus B = \{x \in E^N | (B)_x \subseteq A\} \quad (3.5)$$

Element za strukturiranje *B* se može promatrati i kao uzorak koji se pomiče po slici *A*, prvenstveno po granicama kontura objekata detektiranih na slici *A*, tj. ovaj uzorak provjerava prostornu prirodu skupa *A* po svakoj točki. U točkama u kojima je *B* translatiran za *x* može biti sadržana u skupu *A* (postavljanjem ishodišta skupa *B* na *x*), odnosno kad se element za strukturiranje ne nalazi na rubu oblika koji definira skup A, tada *x* pripada skupu $A \ominus B$.

Erozija slike A elementom za strukturiranje B predstavlja sjecište svih translacija skupa *A* točkama -*b*, gdje je $b \in B$:

$$A \ominus B = \bigcap_{b \in B} (A)_{-b} \quad (3.6)$$

3.6.2. Dilacija

Dilacija se koristi za dodavanje piksela granicama objekata detektiranima na slici, za razliku od erozije koja miče piksele s granica objekata. Prema [37], dilacija je morfološka transformacija koja kombinira dva skupa *A* i *B* korištenjem vektora parova elemenata kako je opisano u nastavku. Ako su *A* i *B* skupovi u Euklidskom *N*-prostoru (E^N) s elementima *a* i *b*, pri čemu su $a = (a_1, \cdots, a_n)$ i $b = (b_1, \cdots, b_n)$ n-torke od elemenata koji predstavljaju koordinate slike, tada dilacija skupa *A* skupom *B* predstavlja skup svih suma vektora parova elemenata, jednog koji dolazi iz skupa *A* i jednog koji dolazi iz skupa

B. Pretpostavkom da su skupovi *A* i *B* podskupovi skupa E^N, dilacija skupa *A* skupom *B* se označava $A \oplus B$ i definiran izrazom:

$$A \oplus B = \{c \in E^N | c = a + b \text{ za } a \in A \text{ i } b \in B\} \tag{3.7}$$

Nadalje, pod pretpostavkom da je skup *A* podskup skupa E^N i $x \in E^N$, tada je translacija skupa *A* za *x*, što se označava s $(A)_x$, definirana izrazom:

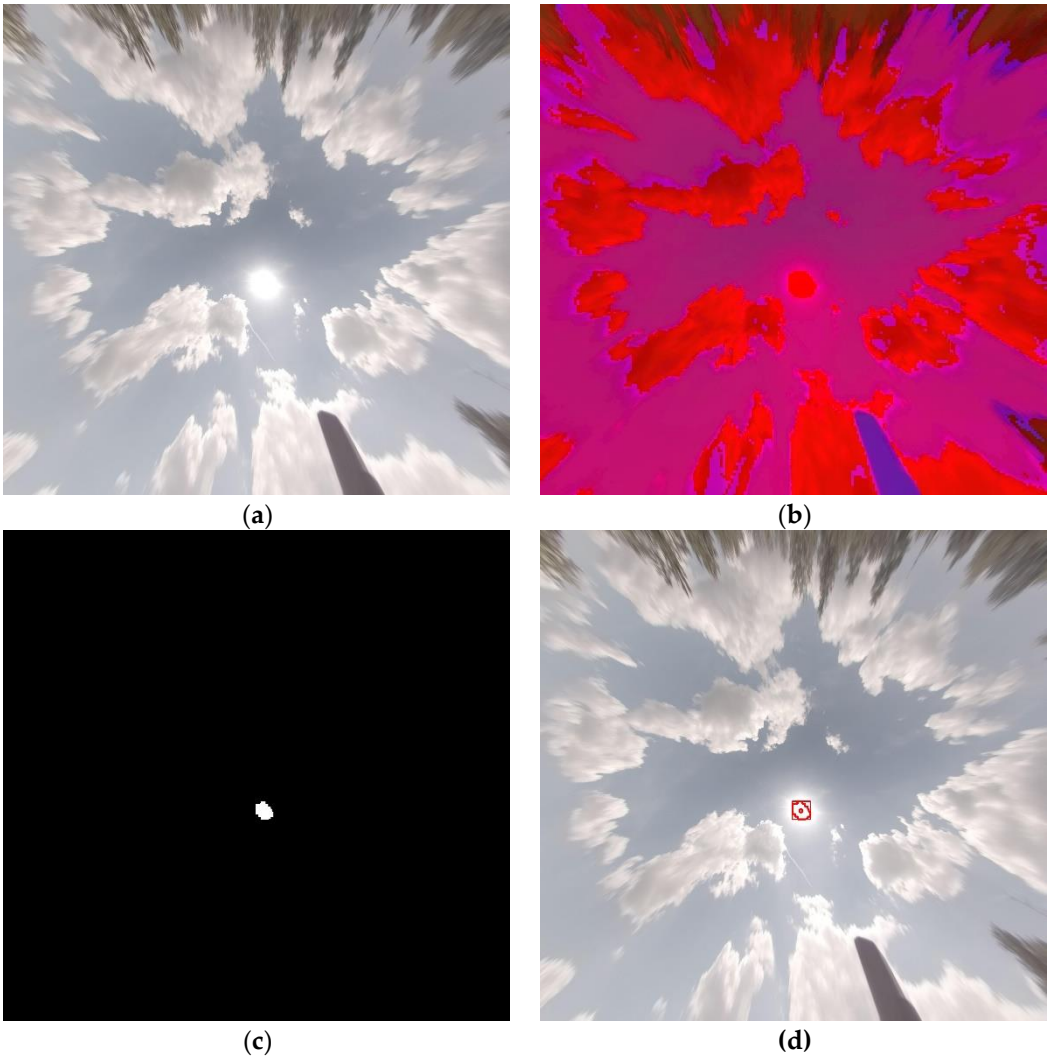
$$(A)_x = \{c \in E^N | c = a + x \text{ za } a \in A\} \tag{3.8}$$

Dilacija skupa A strukturnim elementom B možete se računati kao unija translacija A elementima skupa B:

$$A \oplus B = \bigcup_{b \in B} (A)_b \tag{3.9}$$

3.7. Detekcija kontura oblaka i sunca

Konverzija iz RGB u HSV model boja može se izvršiti korištenjem JavaCV biblioteke opisane u poglavlju 3.2 korištenjem metode „cvtColor“ iz klase „Imgproc“. S obzirom na to da se slike koje prikazuju Sunce sastoje od blještavih segmenata na poziciji gdje se nalazi Sunce, kod konverzije RGB u HSV model boja postavile su se minimalne i maksimalne vrijednosti za H i S komponente na vrijednost 0 (kako bi se „filtrirale“ vrijednosti koje predstavljaju boje, jer se za detekciju sunca potrebno uzeti u obzir samo najblještavije elemente slike), a za V komponentu su se koristile vrijednosti u rasponu između 240 i 255 kako bi se „filtrirali“ samo segmenti koji jako blješte na slici. Na sljedećoj slici prikazan je primjer obrađivanje slike s ciljem detekcije Sunca:



Slika 3.9. Proces obrade slike u svrhu detekcije Sunca: (a) originalna slika, (b) HSV format slike korištenjem minimalnih vrijednosti HSV(0, 0, 240) i maksimalnih vrijednosti HSV(0, 0, 255), detektirano Sunce na slici označeno konturom bijele boje na crnoj pozadini i (d) detektirana kontura Sunca korištenjem aproksimacije oblikom pravokutnika

Strukturni element kod detekcije Sunca prilikom implementacije morfološke operacije erozije bio je postavljen na 50 x 50 piksela. Veličina je određena na temelju rezolucije slike (3104 x 3000 piksela) i kriterija za određivanje najblještavijeg mjesta na slici koji predstavlja Sunce. Slika 3.8. prikazuje algoritam korišten kod obrađivanja slike, a na slici 3.9. prikazani su koraci u detekcije oblika Sunca. Slika 3.9 (a) prikazuje originalnu sliku nakon odstranjivanja efekta distorzije, a slika 3.9 (b) prikazuje konvertiranu sliku iz RGB u HSV model boja definirano „filterom“ HSV komponenti s minimalnim vrijednostima (0, 0, 240) te maksimalnim vrijednostima (0, 0, 255) kako bi

se iz obrađene slike odstranile svi oblici koji nemaju karakteristike sunca (najblještavije područje na slici). Slika 3.9. (c) prikazuje sliku koja nastaje nakon izvršavanja procesa erozije (pri čemu bijeli dio slike predstavlja detektirano Sunce), a slika 3.9. (d) predstavlja originalnu sliku na kojoj je detektirana krivulja Sunca iscrtana crvenom bojom, zajedno s aproksimacijom oblika pravokutnika te centroidom krivulje. Način na koji se određuje centroid krivulje objašnjen je u sljedećem poglavlju. H i S komponente su postavljene na vrijednost 0 radi toga što je kod detekcije Sunca potrebno ignorirati komponente slike u boji (H komponenta) te intenzitet sive boje (predstavljen S komponentom). Vrijednost V predstavlja komponentu kojom se opisuje svjetlina boje i pomoću nje se detektiraju najblještaviji segmenti slike. Ona se jedina koristiti s graničnim vrijednostima od 240 (minimalna vrijednost) i 255 (maksimalna vrijednost) kako bi se postiglo segmentiranje samo najblještavijih tijela na slici, odnosno detektiranje Sunca.

Nakon detektiranja Sunca i korištenja morfološke operacije erozije, jedina kontura koja je detektirana je kontura Sunca. Korištenjem metode „findContours“ metode iz klase „Imgproc“ JavaCV biblioteke, kontura Sunca je precizno detektirana i iscrtana korištenjem metode „drawContours“ na slici 3.9 (d). Nakon toga je korištenjem metode „minAreaRect“ iz klase „Imgproc“ određen aproksimirani oblik pravokutnika s minimalnom površinom koji opisuje oblik detektiranog Sunca te iscrtan na istoj slici.

Nakon uspješne detekcije Sunca, sljedeći korak je detektiranje oblaka na istoj slici. Proces detektiranja rubova oblaka se također sastoji od pretvorbe modela boja iz RGB u HSV, izvršavanjem morfoloških operacija erozije i dilacije te aproksimacije oblika oblaka pravokutnikom koji ima minimalnu površinu s kojom opisuje krivulju svakog od oblaka. Na slici 3.11. je prikazan proces obrade slike neba u svrhu detekcije oblaka.

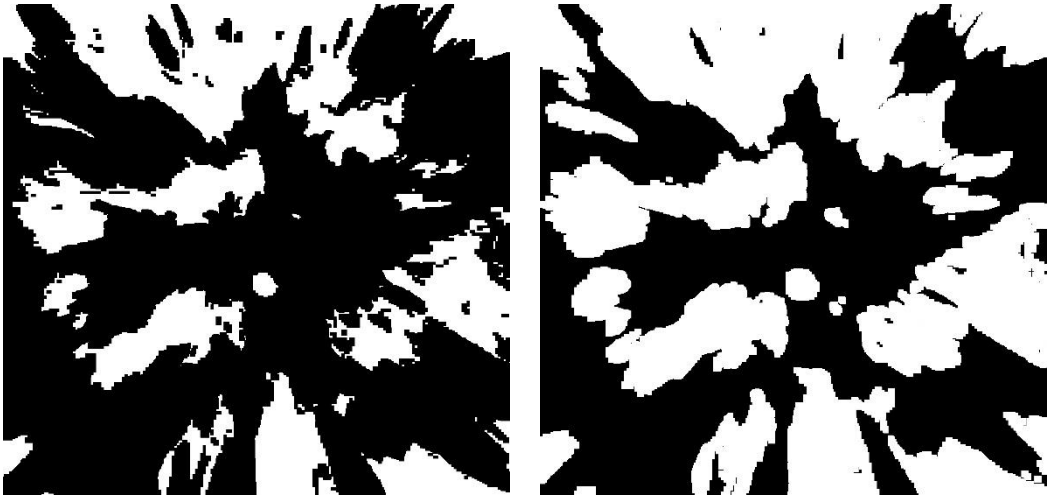
Minimalne i maksimalne vrijednosti za HSV komponente su postavljene na (0, 0, 0) i (70, 70, 255) te su s njima ostvareni najbolji rezultati kod detekcije oblaka na slici neba. Za razliku od parametra za detekciju Sunca kad se na slici tražio samo najblještaviji segment, kod detekcije oblaka je bitno usredotočiti se na ostale boje, a ne samo intenzivno osvjetljene segmente slike, kao što su različite nijanse sive boje koje poprimaju oblaci. H komponenta je postavljena u rasponu od 0 do 70, pri čemu se pokazalo da je u tom području dominantna boja crvena (što je prikazano na slici 3.10 (b)). S komponenta

opisuje intenzitet sive boje u boji na slici pa reduciranje vrijednosti te komponente rezultira transformacijom koja na obrađenoj slici naglašava sivu boju, koja je vrlo često prisutna na samim oblacima. V komponenta nije filtrirana u slučaju detekcije oblaka i uključen je cijeli raspon vrijednosti od 0 do 255 te u kombinaciji sa S komponentom opisuje svjetlinu boje kako bi se obuhvatile sve nijanse sive boje. Originalna slika prikazana je na slici 3.10 (a), a transformiranja slika nastala korištenjem opisanog HSV filtra je prikazana na slici 3.10 (b).

Veličina strukturnog elementa za morfološku operaciju erozije je postavljena na veličinu od 20x20 piksela i određena je rezolucijom slike 3104x3000 piksela. Tom veličinom se postiže odstranjivanje vrlo malih oblaka sa slike nema koji ne bi mogli značajnije prekriti Sunce.

(**a**)

(**b**)



(c) (d)

Slika 3.10. Proces obrade slike u svrhu detekcije oblaka: (a) originalna slika, (b) slika u HSV modelu boja, (c) detektiranje krivulja oblaka nakon izvršavanja operacije erozije i (d) detektirane krivulje korištenjem operacije dilacije

Površina ispod koje su oblaci smatrani premalima je postavljena na 3000 piksela. Rezultirajuća slika nakon transformacije korištenjem morfološke operacije erozije prikazana je na slici 3.10 (c). Može se primijetiti da su konture oblaka smanjene u odnosu na originalnu sliku, kao što su i maknuti vrlo mali oblaci sa slike. Kako bi se ispunile nesavršenosti detektiranih krivulja i oblaci poprimili vjerodostojniji izgled, nakon operacije erozije potrebno je izvršiti morfološku operaciju dilacije. Veličina strukturnog elementa za tu svrhu bila je postavljena na 30x30 piksela. Veličina je određena na temelju rezolucije slike 3104x3000 piksela te kriterija s kojim su se dobili najbolji rezultati u pogledu popunjavanja manjih rupica na površini objekata koje nastaju korištenjem morfološke operacije erozije. U slučaju detekcije Sunca korištenje operacije dilacije nije bilo potrebno, već samo erozije, jer najblještaviji segment slike je uvijek popunjen i nije ga potrebno proširiti, već naprotiv, odstraniti one segmente slike koji sigurno ne predstavljaju segmente Sunca. Rezultati transformacije nakon operacije dilacije i detekcije oblaka mogu se vidjeti na slici 3.10 (d).

3.8. Detekcija centroida oblaka i Sunca

Nakon detekcije Sunca i oblaka te korištenjem operacija erozije i dilacije, sve je spremno za određivanje centroida krivulja koje predstavljaju Sunce i oblake. Korištenjem

metode „findContours“ iz klase „Imgproc“ JavaCV biblioteke je moguće pronaći sve konture na slici neba, a nakon toga ih je moguće iscrtati na slici korištenjem metode „drawContours“. Primjer iscrtavanja krivulja Sunca crvenom bojom prikazan je na slici 3.9 (d). Osim same krivulje na slici je prikazan je aproksimacija oblika Sunca oblikom pravokutnika koji zauzima minimalnu površinu, a da i dalje opisuje površinu Sunca. Taj aproksimirani oblik pravokutnika je moguće odrediti pomoću metode „minAreaRect“ iz klase „ImgProc“ te naknadno iscrtati korištenjem metoda za iscrtavanje linija „line“.

Kako se oblaci i Sunce kreću po horizontu neba, za predviđanje njihovog kretanja je potrebno odrediti njihove centroide. Sunce se kreće po nebu dosta sporije od oblaka pa je potrebno pratiti njegovo kretanje duže od oblaka. Kako se kamerom slika nebo svakih pet sekundi, pomaci u tom vremenskom razdoblju su dosta mali, ali ako se uzme više takvih vremenskih perioda u obzir, na primjer 12 slika, što rezultira promjenom pozicije oblaka unutar jedne minute ili nekoliko minuta, tad je predviđanje gibanja točnije. Koordinate centroida krivulje mogu se odrediti korištenjem sljedećeg izraza:

$$c = \frac{1}{n}\sum_{i=0}^{n}(x_i, y_i) \tag{3.10}$$

gdje je *c* centroid Sunca, *n* broj točaka u detektiranoj krivulji Sunca ili oblaka, a (x_i, y_i) koordinate točaka u detektiranoj krivulji Sunca ili oblaka.

Za pronalaženje centroida sunca ili oblaka korištenjem JavaCV biblioteke, potrebno je koristiti momente. Moment je srednja vrijednost težine intenziteta piksela slike koji se koriste kako bi se odredila specifična svojstva slike, kao što je centroid krivulje. U slučaju rasterske slike u obliku pravokutne mreže točaka, prostorni momenti su izračunati prema izrazu 3.11 [38]. Središnji momenti izračunavaju se prema izrazu 3.12, gdje je $(\acute{x}, \acute{y})$ centroid određen izrazima 3.13 i 3.14.

$$m_{ji} = \sum_{x,y}\left(array(x,y)\cdot x^j \cdot y^j\right) \tag{3.11}$$

$$mu_{ji} = \sum_{x,y}\left(array(x,y)\cdot (x-\bar{x})^j \cdot (y-\bar{y})^j\right) \tag{3.12}$$

$$\acute{x} = \frac{m_{10}}{m_{00}} \tag{3.13}$$

$$\acute{y} = \frac{m_{01}}{m_{00}} \tag{3.14}$$

gdje m_{ji} predstavlja prostorni moment, $\boldsymbol{array}(\boldsymbol{x}, \boldsymbol{y})$ polje dvodimenzionalnih točaka na slici, x^j i y^jkoordinate točaka, mu_{ji} središnje momente, a $(\acute{x}, \acute{y})$ predstavlja centar mase. Slika 3.9 (d) prikazuje primjer iscrtanog centroida Sunca u obliku crvene točke koja se nalazi unutar krivulje sunca.

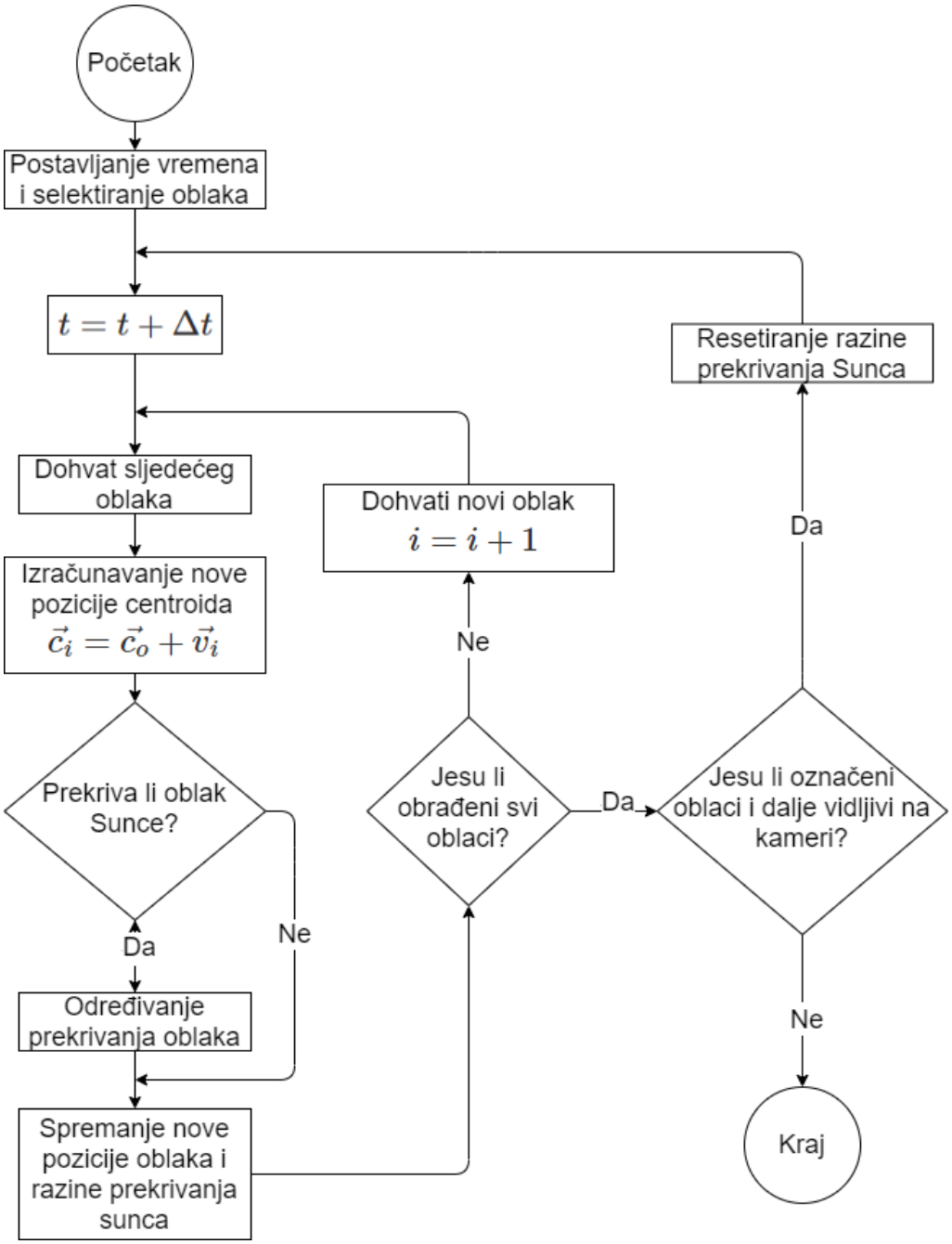
4. Predviđanje zasjenjenja sunca

Zaklanjanje sunca oblacima smanjuje razinu dozračenosti, a nastaje kad oblaci prekriju površinu sunca. Da bi se odredio trenutak zasjenjenja Sunca, potrebno je odrediti trenutak kad će površina oblaka prekriti površinu sunca. Trenutak početka zasjenjenja Sunca određuje se na način da se odredi pomak centroida oblaka, uzevši u obzir i pomak Sunca. Pomakom centroida oblaka i Sunca određuje se i pomak cijele njihove konture. Kontura oblaka i sunca aproksimira se oblikom pravokutnika kako bi se pomak konture pojednostavnio, tj. sveo samo na pomak četiri točke koje tvore pravokutnik najmanje površine koji opisuje konturu. S obzirom da su konture sunca i svih oblaka aproksimirane oblikom pravokutnika, međusobno prekrivanje pravokutnika određuje se korištenjem metode „rotatedRectangleIntersection“ iz klase „Imgproc“. Postoje tri moguća načina preklapanja pravokutnika: (i) u potpunosti (što se označava konstantnom vrijednošću „Imgproc.INTERSECT_FULL“), (ii) djelomično (što se označava konstantnom vrijednošću „Imgproc.INTERSECT_PARTIAL“) ili (iii) bez prekrivanja (označeno konstantnom vrijednošću „Imgproc.INTERSECT_NONE“). Razine zaklanjanja sunca u svakom trenutku obrade slike spremaju se u CSV datoteku na temelju koje se predviđa razina dozračenosti u budućnosti.

Za određivanje pomaka centroida uzima se 12 uzastopnih slika te se računa razlika u koordinatama centroida na prvoj slici u tom nizu i na 12. slici. S obzirom da samo jedan pomak centroida unutar pet sekundi (između dvije slike) može biti nemjerodavan (rezultat spajanja dva različita oblaka što rezultira sa znatnim pomakom dva centroida od prethodna dva oblaka koji se spajaju u jedan), ukupan pomak za 12 slika predstavlja temelj za određivanje nove pozicije centroida u narednom periodu. S obzirom da kamera slika nebo svakih pet sekundi, za 12 slika prođe jedna minuta. Prema tom pomaku unutar minute se određuju i ostali pomaci oblaka i Sunca. S obzirom da se na svakoj slici ponovno određuju pozicije centroida svih kontura oblaka i Sunca, kod određivanja nove pozicije oblaka svake od konture određuje se najbliži centroid trenutnom i prema tome određuje pomak. U sljedećem poglavlju prikazan je dijagram toka algoritma koji se koristi za određivanje razine zasjenjenja Sunca.

4.1. Algoritam predviđanja zasjenjenja Sunca

Na slici 4.1. prikazan je dijagram toka koji opisuje algoritam koji se koristi za predviđanje zasjenjenja Sunca. Prije početka rada algoritma obavlja se analiza prvih 12 slika neba kako bi se odredili vektori smjera i brzine kretanja oblaka i Sunca (što je prikazano dijagramom toka algoritma na slici 3.8.).



Slika 4.1. Dijagram toka algoritma za predviđanje prekrivanja Sunca oblacima

Algoritam započinje analizom koji oblak će u sljedećem periodu (u sljedećoj minuti) prekrivati sunce. Kad se na svakoj slici detektiraju svi oblaci i njihovi centroidi te pripadajući pomaci, mogu se simulirati i nove pozicije oblaka u sljedećem periodu. Za svaki oblak se određuje simulirana lokacija oblaka u budućnosti aproksimirana oblikom pravokutnika te provjerava prekriva li oblik sunca koji je također aproksimiran pravokutnikom. Razina zasjenjenja određuje se pozivom metode „rotatedRectangleIntersection“ iz klase „Imgproc“ te se podaci spremaju u objekt klase „CoverageRecord“ te sprema u „ArrayList“ strukturu, što je prikazano u Java implementaciji u Prilogu C.

Dijagram toka na slici 4.1. počinje s iniciranjem početka mjerenja vremena i postavljanja varijable $\mathbf{t = 0}$, odabirom prvog oblaka s indeksom $\boldsymbol{i}$, te postavljanjem brojača $\boldsymbol{k}$ na vrijednost 0. Vanjska petlja inkrementira brojač, vrijeme $\boldsymbol{t(k)}$ za $\mathbf{\Delta t}$ te izračunava nove pozicije centroida s indeksom $\boldsymbol{i}$:

$$\vec{\boldsymbol{c}}_{\boldsymbol{i,k+1}} = \vec{\boldsymbol{c}}_{\boldsymbol{i,k}} + \vec{\boldsymbol{v}}_{\boldsymbol{i,k}}\,\mathbf{\Delta t} \qquad (4.1)$$
$$\boldsymbol{t(k)} = \boldsymbol{t(k-1)} + \mathbf{\Delta t}$$

gdje $\vec{\boldsymbol{c}}_{\boldsymbol{i,k}}$ predstavlja vektor gibanja oblaka $\boldsymbol{i}$ u trenutačno definiranom brojačem $\boldsymbol{k}$, $\vec{\boldsymbol{c}}_{\boldsymbol{i,k+1}}$predstavlja vektor gibanja oblaka i u trenutku definiranom u sljedećem periodu (nakon pet sekundi) $\boldsymbol{k+1}$, $\vec{\boldsymbol{v}}_{\boldsymbol{i,k}}$ predstavlja vektor brzine, $\mathbf{\Delta t}$ vremenski period od 5 sekundi, $\boldsymbol{t(k)}$ vrijeme nakon $\boldsymbol{k}$ perioda, a $\boldsymbol{t(k-1)}$vrijeme u prethodnom periodu.

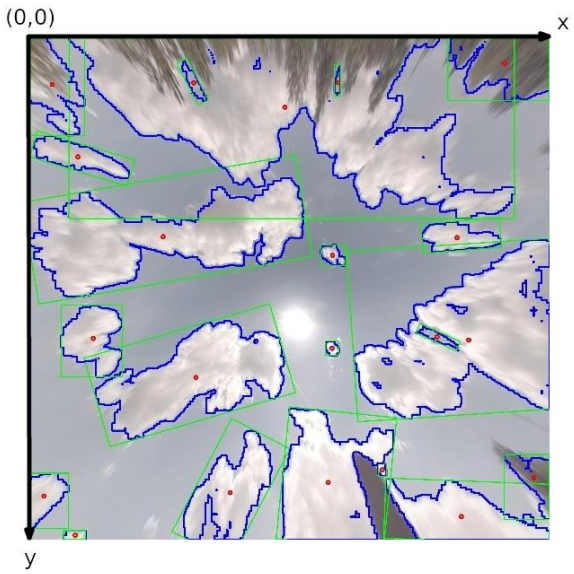
Cilj algoritma je odrediti stupanj pokrivenosti sunca za sljedećih sat vremena. Ovaj signal definiraju dva polja $\boldsymbol{t}$ i $\boldsymbol{pokrivenost}$ koja zajedno predstavljaju diskretni oblik stupnja pokrivenosti tijekom vremena. U tu svrhu uspostavljena su dva iterativna procesa: (i) unutarnja petlja koja izračunava solarnu dozračenost u određenom trenutku procjenom položaja svih oblaka za taj trenutak i izračunavanjem ukupne pokrivenosti sunca i (ii) vanjska petlja koja određuje promjenu u vremenu za dobivanje položaja i pokrivenosti sunca za sve trenutke, tj. za izračunavanje vrijednosti $\boldsymbol{pokrivenost}\mathbf{(t)}$.

Algoritam se izvršava toliko dugo dok su prisutni oblaci koji su bili prisutni na slici neba na samom početku (tijekom analize prvih 12 slika), tj. sve dok ne napuste područje

slike. To se određuje na način da se promatra lijevi gornji kut pravokutnika koji aproksimira oblik krivulje te provjerava je li napustio područje slike definirano rezolucijom i ishodištem s koordinatama (0, 0) kao što je prikazano na slici 4.2.

4.2. Aproksimacija oblika oblaka i Sunca pravokutnikom

Radi optimiziranja vremena potrebnog za simuliranje pomaka oblaka u budućnosti i provjeravanje prekriva li površinu Sunca, umjesto detektirane konture koristi se aproksimacija oblika pravokutnikom. Aproksimacija oblika pravokutnikom opisana je klasom „RotatedRect“ koju vraća metoda „minAreaRect“ iz klase „Imgproc“. Ta metoda prima skup točaka koje opisuju zatvorenu konturu, u ovom slučaju konturu oblaka i Sunca. Na slici 4.2. prikazana je slika neba koja sadržava označene konture oblaka označenih plavom bojom, a točke koje označavaju centroide su označene crvenom bojom. Aproksimirani oblici oblaka pravokutnicima označeni su zelenom bojom. Kontura Sunca nije označena jer je nakon detekcije Sunca u prvoj fazi algoritma obrade ta krivulja spremljena i kad je ta ista krivulja detektirana kod detekcija oblaka (ili krivulja koja je vrlo blizu te konture, manje od 100 piksela udaljenosti), onda se onda ignorira kod obrade slike.



Slika 4.2. Prikaz oblika oblaka aproksimiranih oblikom pravokutnika

Razina zasjenjenja određuje se pozivom metode „rotatedRectangleIntersection“ iz klase „Imgproc“ te se podaci spremaju u objekt klase „CoverageRecord“ te sprema u „ArrayList“ strukturu. Ti podaci se na kraju koriste za kreiranje CSV datoteke koja

sadržava podatke o zasjenjenju Sunca za svaki od oblaka u svakom trenutku vremena sve dok je simulirana lokacija oblaka u budućnosti i dalje prisutna na slici neba koja se obrađuje. Postotak zasjenjenja Sunca određuje se tako da se prema površini pravokutnika koji aproksimira oblik Sunca u pikselima određuje broj piksela koji se preklapaju s pravokutnikom koji aproksimira oblik oblaka. Dijeljenjem površine preklapanja pravokutnika s ukupnom površinom sunca dobiva se postotak prekrivanja sunca oblacima.

5. Predviđanje sunčeve dozračenosti

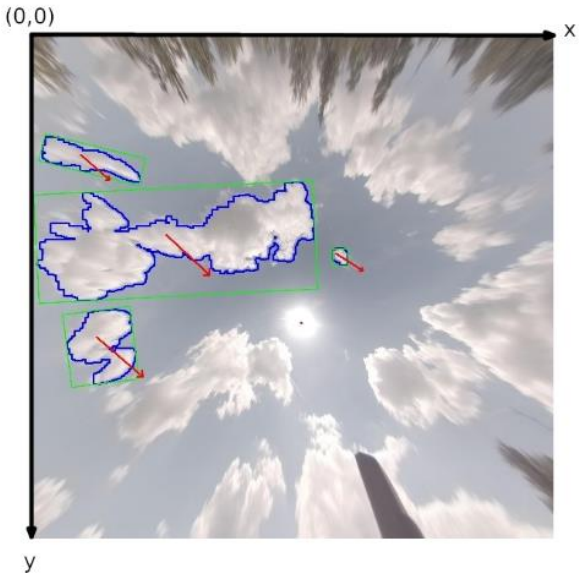
Detektirane konture oblaka s točkama centroida i aproksimiranim pravokutnicima tvore polazišnu točku za predviđanje sunčeve dozračenosti u ovisnosti o vidljivoj površini sunca koja nije prekrivena oblacima.

Određivanjem vektora brzine i smjera analizom 12 uzastopnih slika pri čemu je svaka slikana u razmaku od pet sekundi dobiva se pomak centroida oblaka i Sunca u tom periodu. Taj pomak koristi se kako bi se simulirao pomak cijelog oblaka i sunca, tj. aproksimacije njegovog oblika pravokutnikom. S obzirom da na slici neba postoje oblaci koji će sigurno zasjeniti Sunce, oblaci koji će vjerojatno zasjeniti Sunce te oni koji sigurno više neće zakloniti Sunce (jer se njihova pozicija i kretanje na nebu udaljava od sunca ili je njihova putanja predaleko od sunca), u obzir se uzimaju oblaci koji će sigurno ili vjerojatno zasjeniti Sunce. Time se postižu poboljšane performanse sustava te analiza i predikcija zasjenjenja Sunca za sljedeći period obavlja se u samo nekoliko sekundi.

Korištenjem OpenCV funkcija je moguće odrediti preklapanje površine dvaju pravokutnika koji aproksimiraju oblike oblaka i Sunca u pikselima, a kako je u prethodnim koracima algoritma određena površina Sunca, može se u postocima odrediti razina zasjenjenja Sunca. Dobivena razina zasjenjenosti se kombinira s predviđenom srednjom razinom direktne dozračenosti od strane meteorološkog servisa pa je moguće dobiti krivulju koja prikazuje točnu razinu direktne dozračenosti u sljedećem periodu od sat vremena što će biti ulazni parametar MATLAB/Simulink modelu za optimiranje tokova energije.

5.1. Određivanje kandidata oblaka za zaklanjanje Sunca

Slika 5.1. prikazuje sliku neba na kojoj su označeni samo oblaci koji su kandidati za prekrivanje Sunca u budućem periodu. Na temelju pomaka centroida oblaka prikazane su i strelice koje prikazuju vektor brzine i smjera oblaka, a duljina vektora prikazuje koliko se oblaci pomiču za pet minuta.



Slika 5.1. Prikaz neba s oblacima koji su kandidati za prekrivanje Sunca

Kandidati oblaci koji mogu zasjenjivati Sunce određeni su na temelju njihove pozicije i pozicije sunca, tj. njihovih centroida. Ako su se oblaci već udaljili od Sunca (te više ne mogu prekriti Sunce) ili ako su previše udaljeni od Sunca (simuliranim gibanjem ustanovljeno da u budućnosti ne mogu biti blizu Sunca i ne mogu ga prekriti), oni se ne prate kako bi se dodatno optimiziralo trajanje izvođenja algoritma. Na slici 5.1. su označeni sami četiri oblaka koji su u blizini Sunca i mogli bi prekrivati Sunce u budućnosti.

5.2. Određivanje vektora brzine oblaka i sunca

Niz od 12 početnih slika koristi se kako bi se odredio vektor brzine kojom se definira kretanje oblaka i sunca te predviđa njihovo kretanje u budućnosti. Vrijeme

potrebno da kamera snimi 12 slika je 60 sekundi (12 x 5 sekundi). Promjena pozicije i predikcija kretanja oblaka i sunca se temelji na kretanju tijekom posljednje minute. Slika 4.1. prikazuje dijagram toka koji opisuje na koji način se određuje razina zasjenjenja Sunca pomoću opisanog algoritma. S obzirom da je pozicija centroida Sunca i oblaka određena s algoritmom opisanom na slici 3.8, vektor brzine oblaka se određuje na temelju pomaka centroida unutar posljednjih 60 sekundi. Koliko se pomaknu oblaci unutar pet minuta, prikazano je na slici 5.1. S obzirom da se Sunce kreće sporije od oblaka, korištenjem istog algoritma, samo za konturu Sunca, na slici 5.2. je prikazan vektor pomaka sunca unutar sat vremena. Crvena strelica prikazuje koliko se pomaknuo centroid krivulje koja predstavlja konturu Sunca, a crveni pravokutnik prikazuje aproksimirani oblik Sunca.

Slika 5.2. Slika neba na kojoj je prikazan pomak sunca unutar sat vremena

Kod detektiranja najbližeg centroida krivulje na slici neba nakon razlike vremena Δt (60 sekundi), koristi se najbliži centroid i izraz za računanje udaljenosti između dvije točke u koordinatnom sustavu koji određuje najbliži centroid od krivulja koje predstavljaju sve ostale oblake na nebu $d(c_2, c_1)$:

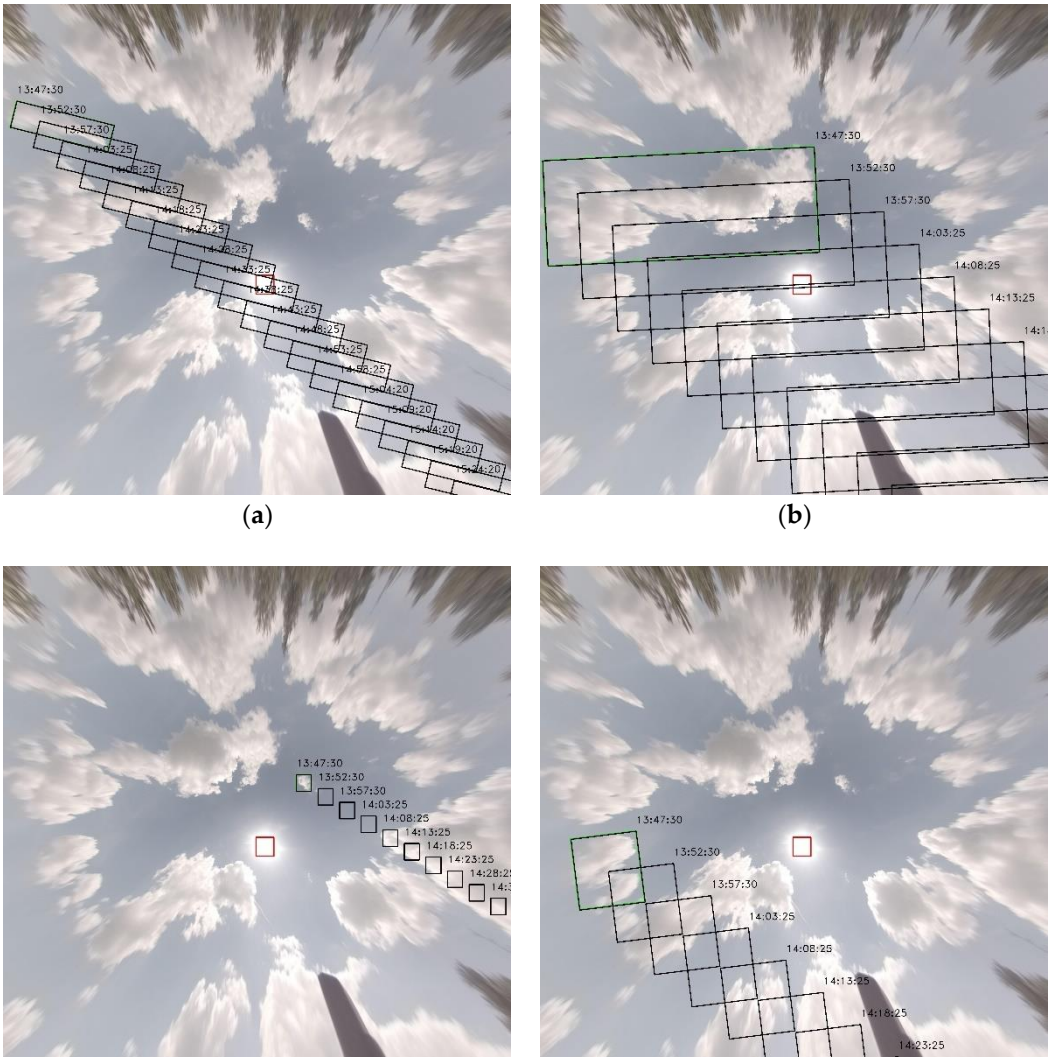
$$d(c_2, c_1) = \sqrt{\left(x_{c_2} - x_{c_1}\right)^2 + \left(y_{c_2} - y_{c_1}\right)^2} \tag{5.1}$$
$$\overrightarrow{v_l} = (x_{c_2} - x_{c_1})\vec{\imath} + (y_{c_2} - y_{c_1})\vec{\jmath}$$

gdje c_1 i c_2 predstavljaju dva najbliža centroida, x_{c_1} i x_{c_2}x koordinate centroida, y_{c_1} i y_{c_2} y koordinate centroida, a $\overrightarrow{v_l}$ vektor brzine. Vektor brzine određuje se pomoću pomaka točaka centroida po x i y osi nakon vremena $\boldsymbol{\Delta t}$ i množenjem s jediničnim vektorima $\vec{\imath}$ i $\vec{\jmath}$.

5.3. Simuliranje gibanja oblaka

Postupak određivanja ukupne zasjenjenosti Sunca u određenom trenutku u unutarnjoj petlji sastoji se od nekoliko koraka. U prvom koraku odabire se prvi oblak pogodan za zasjenjenje Sunca i izračunava se novi položaj korištenjem njegovog prethodnog položaja i brzine prema jednadžbi (4.1). Prema novom položaju Sunca i oblaka izračunava se stupanj zasjenjenja sunca, a novi položaj oblaka i Sunca te stupanj pokrivenosti za taj oblak i vrijeme se pohranjuju. Tada se odabire sljedeći oblak za procjenu stupnja zasjenjenja. Nakon završetka analize svih oblaka, konačna solarna pokrivenost određuje se kao zbroj pokrivenosti svih odabranih oblaka za određeno vrijeme.

Slika 5.3. prikazuje simulirano kretanje oblaka aproksimirano oblikom pravokutnika za svakih pet sekundi u budućnosti. Za svaki od simuliranih položaja oblaka prisutnih na slici određuje se stupanj zasjenjenja sunca. Prikupljeni parametri korišteni su za simulaciju kretanja oblaka u sljedećem razdoblju. Konture oblaka i Sunca aproksimirane su pravokutnicima koji zauzimaju minimalnu površinu koja i dalje opisuje detektiranu konturu. Svaki pravokutnik oblaka označen je vremenskom oznakom koja predstavlja položaj oblaka u to vrijeme. Sjecište između oblaka i Sunca određuje se metodom "rotatedRectangleIntersection" iz klase "Imgproc". Ako oblaci pokrivaju cijelo Sunce, tada je pokrivenost postavljena na 100% (ukupna površina Sunca u pikselima rezultat je poziva metode), inače je manja od 100%.



(**a**) (**b**)

(**c**) (**d**)

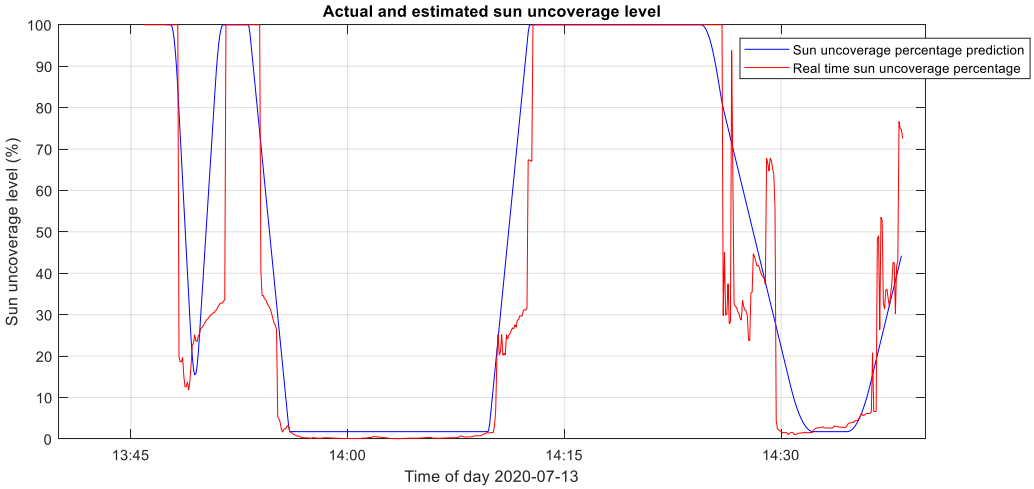
Slika 5.3. Procijenjene pozicije oblaka temeljene na simulacijama kretanja

5.3. Rezultati mjerenja zasjenjenja sunca

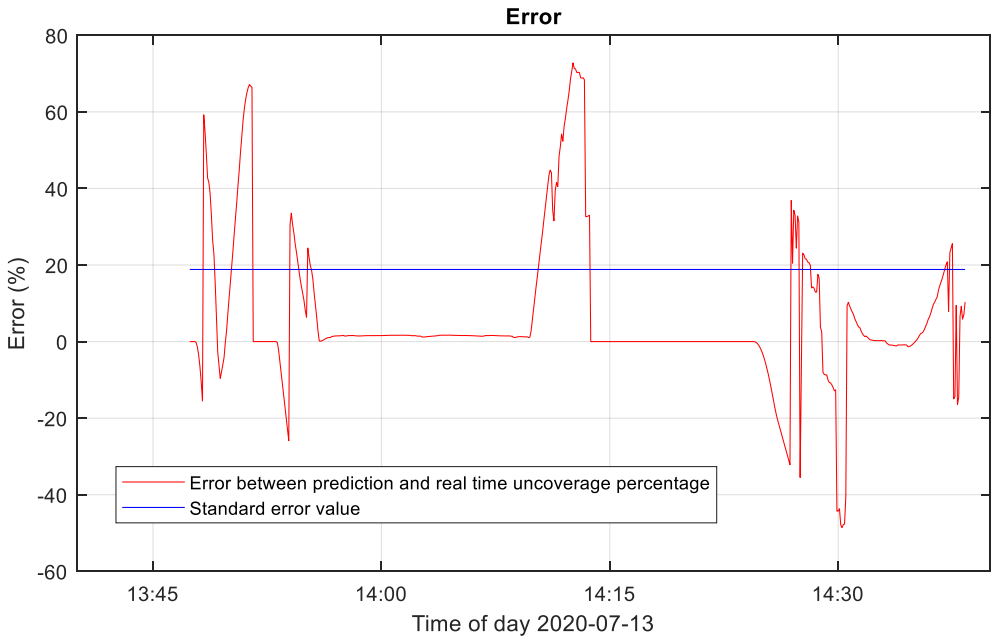
Nakon simulacije gibanja oblaka, određuje se razina zasjenjenja sunca za svaki vremenski pomak od pet minuta toliko dugo dok su oblaci prisutni na nebu slike na kojoj je odstranjen efekt distorzije. Usporedbe trenutačne i predviđene sunčeve dozračenosti prikazane su na slikama 5.4, 5.6, 5.8 i 5.10. Prikazani podaci se odnose na dane 13. 07. 2020. i 14. 07. 2020. u različitim periodima dana. Na njima je prikazana situacija na nebu gdje se oblaci gibaju prema Suncu. Crvena krivulja, koja predstavlja razinu nepokrivenosti Sunca u stvarnom vremenu, pokazuje da se vidljivo područje Sunca značajno mijenja na nelinearan način kako oblaci počinju prekrivati ili otkrivati sunce. Ovaj je učinak uzrokovan prozirnom prirodom oblaka, koja sprječava da se Sunce

potpuno pokrije na početku postupka pokrivanja oblakom, dok aproksimacija pravokutnog oblika taj učinak ne uzima u obzir. Razina nepokrivenosti suncem u stvarnom vremenu temelji se na vidljivoj površini sunca kada je prekriveno oblacima u usporedbi s površinom sunca kad nije prekriveno oblacima.

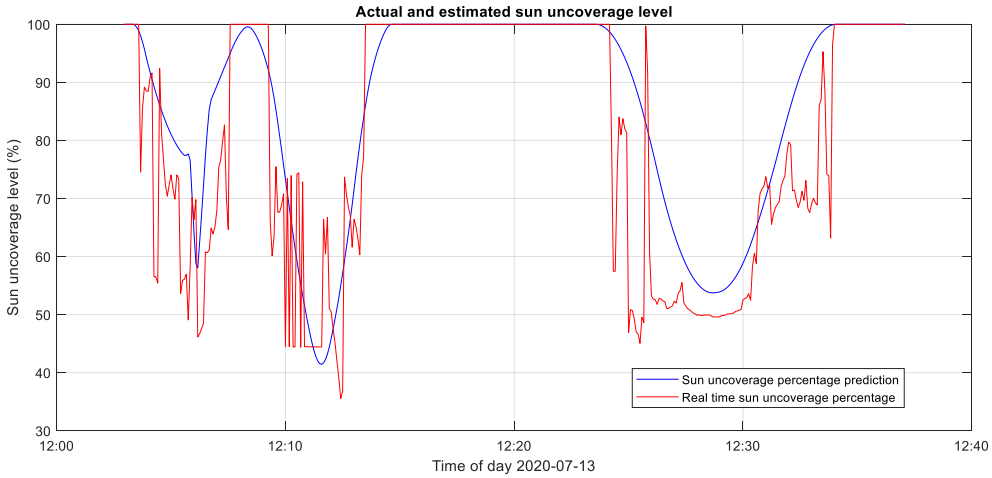
Predviđena razina vidljivosti Sunca, na temelju aproksimacije pravokutnim oblikom dobivenim prethodno opisanim algoritmom, mijenja se na linearni način kao što je prikazano plavom krivuljom na slikama 5.4, 5.6, 5.8 i 5.10, uz odgovarajuće predviđanje razine pogrešaka prikazane na slikama 5.5, 5.7, 5.9 i 5.11. Plava krivulja na slikama 5.4, 5.6, 5.8 i 5.10 predstavlja razinu predviđene sunčeve dozračenosti u postocima prema maksimalnoj vrijednosti u zadanom vremenskom periodu. Predviđanje je izvedeno na početku razdoblja od jednog sata: na slici 5.4. u 13:47h, na slici 5.7 u 12:03h, na slici 5.8 u 10:34h i na slici 5.10 u 16:38h. Nakon što je algoritam analizirao promjenu položaja oblaka i sunca tijekom posljednje minute (korišteno je 12 slika, svaka slika svakih pet sekundi), predviđanje se izvodilo za sljedeći sat. Predviđanje (plava krivulja) uspoređeno je s razinom nepokrivenosti u stvarnom vremenu (crvena krivulja) koja je izmjerena nakon što je predviđanje izvedeno za izračunavanje razine pogreške. Krivulje u stvarnom vremenu i predviđanja se podudaraju u kritičnim područjima gdje je sunce potpuno pokriveno ili potpuno izloženo.



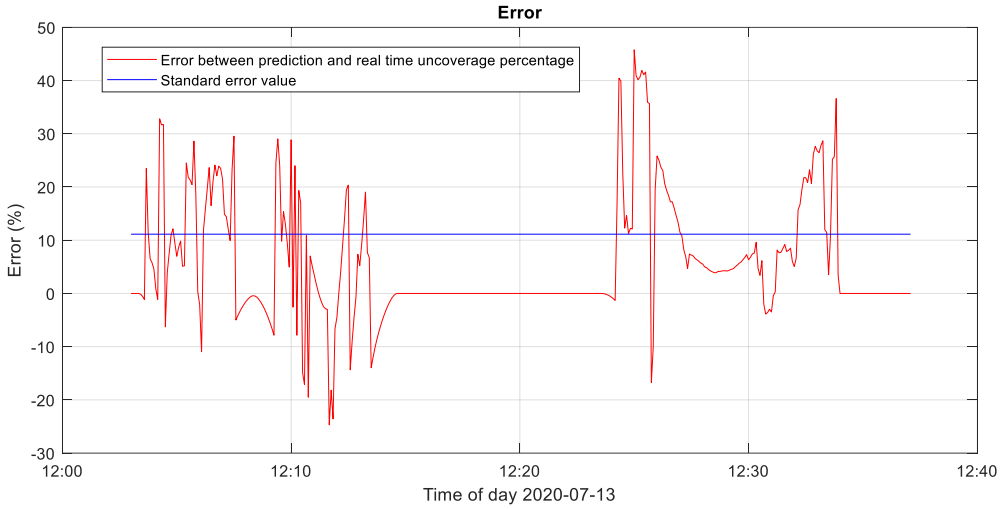
Slika 5.4. Usporedba stvarne i predviđene razine vidljivosti Sunca na dan 13. 07. 2020. između 13:45h i 14:40h.



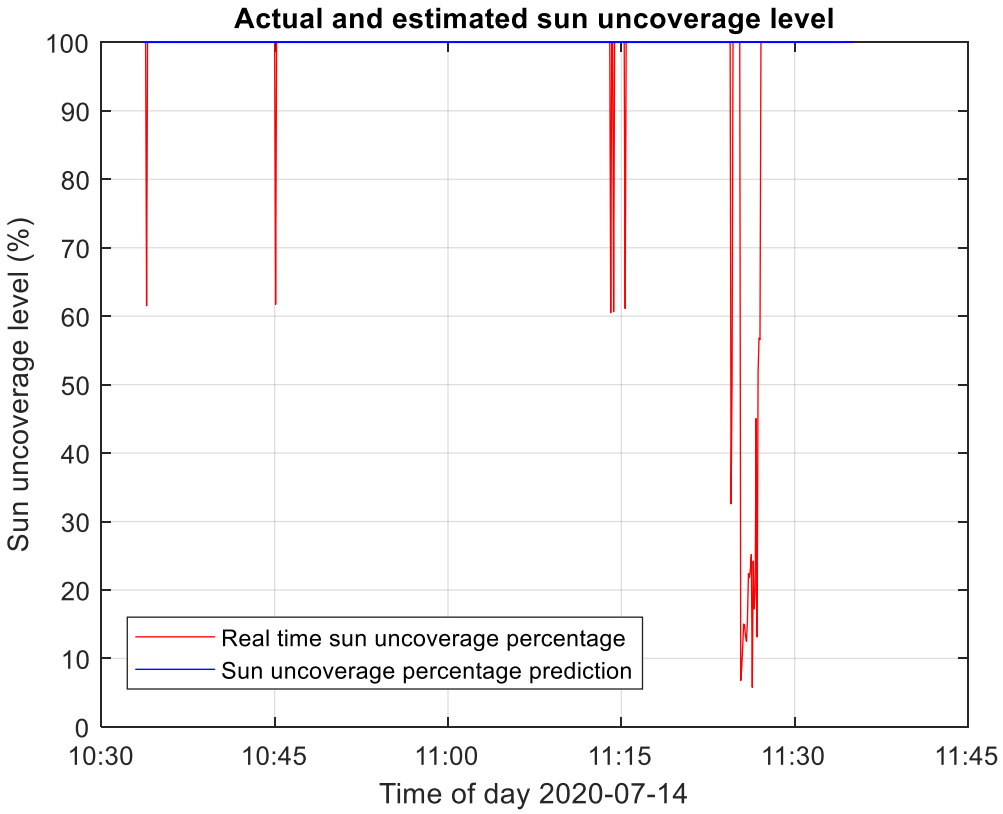
Slika 5.5. Razina pogreške između stvarne i predviđene razine vidljivosti Sunca na dan 13. 07. 2020. između 13:45h i 14:40h.



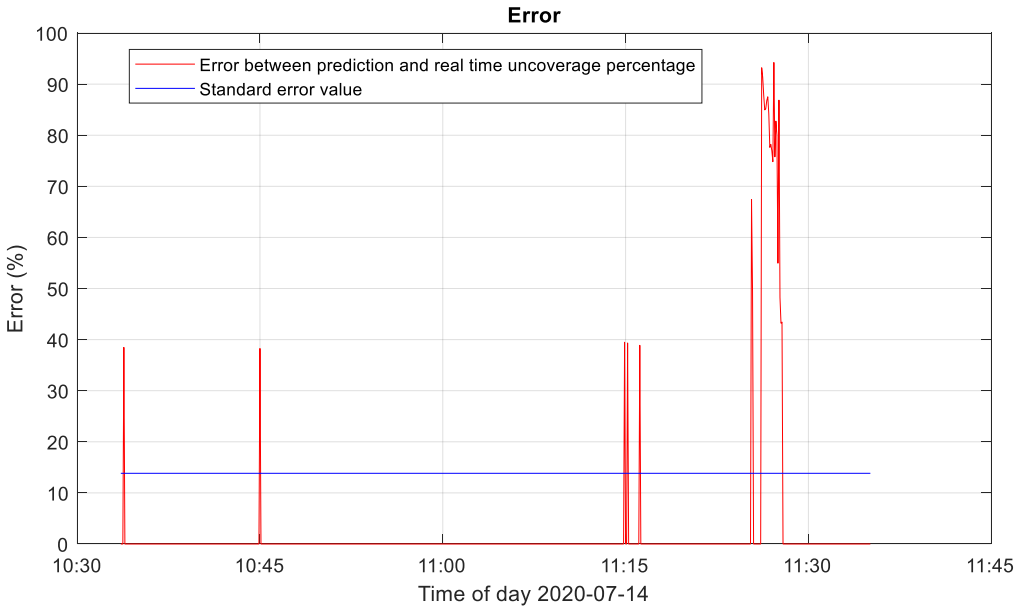
Slika 5.6. Usporedba stvarne i predviđene razine vidljivosti Sunca na dan 13. 07. 2020. između 12:00h i 12:40h.



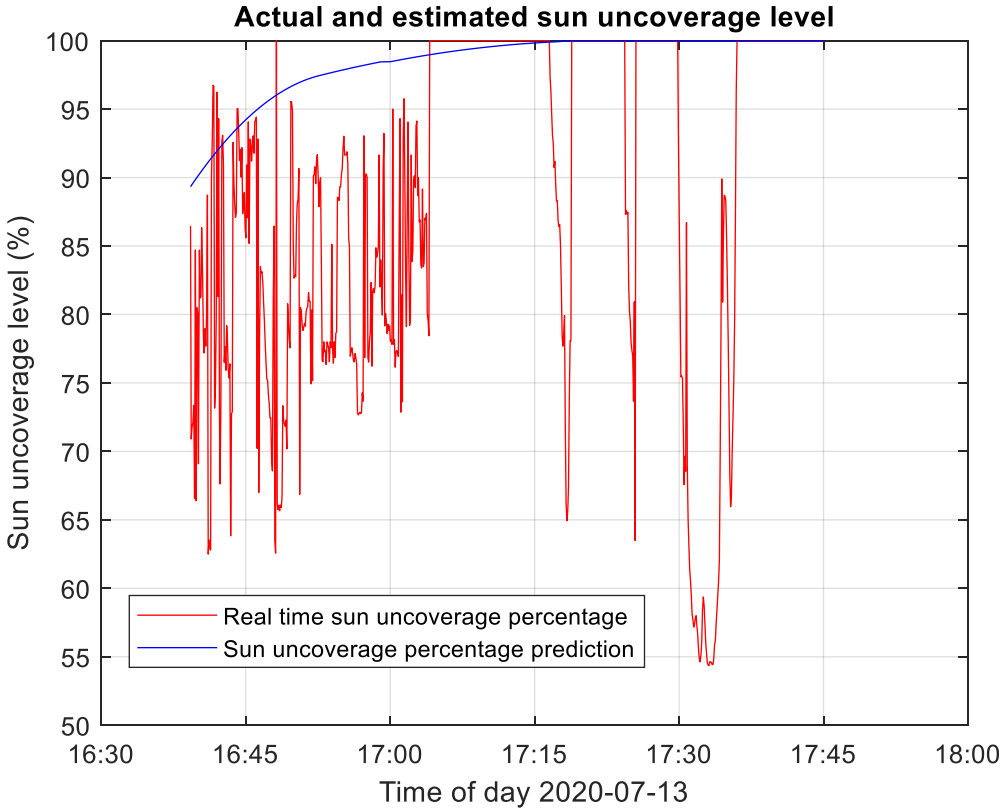
Slika 5.7. Razina pogreške između stvarne i predviđene razine vidljivosti Sunca na dan 13. 07. 2020. između 13:45h i 14:40h.



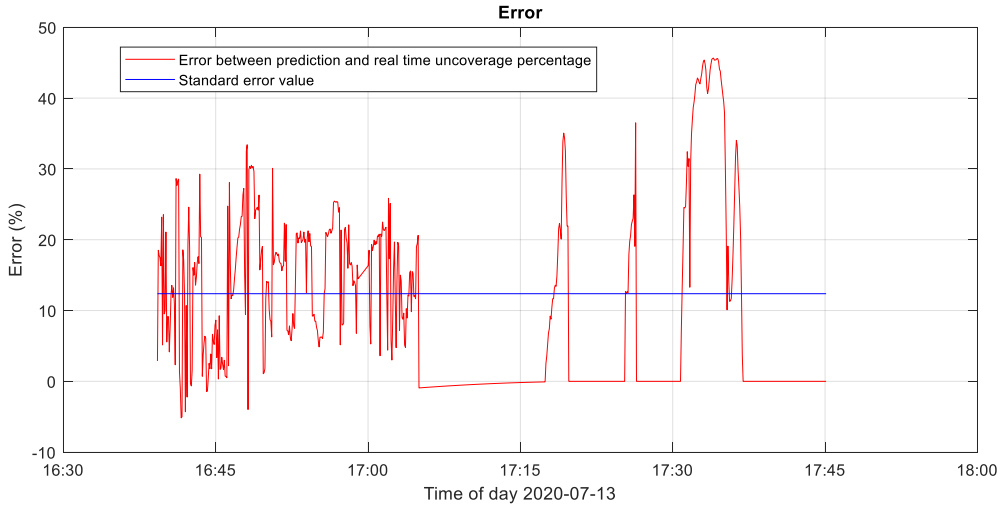
Slika 5.8. Usporedba stvarne i predviđene vidljivosti Sunca na dan 14. 07. 2020. između 10:30h i 11:45h.



Slika 5.9. Razina pogreške između stvarne i predviđene razine vidljivosti Sunca na dan 14. 07. 2020. između 10:30h i 11:45h.



Slika 5.10. Usporedba stvarne i predviđene razine vidljivosti Sunca na dan 13. 07. 2020. između 16:30h i 18:00h.



Slika 5.11. Razina pogreške između stvarne i predviđene vidljivosti Sunca na dan 14. 07. 2020. između 10:30h i 11:45h.

Dobivena krivulja koristi se za predviđanje razine direktne sunčeve dozračenosti izračunate prema sljedećem izrazu:

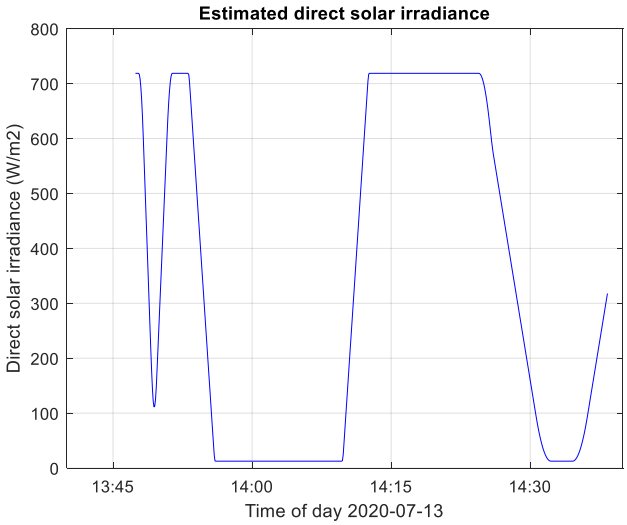
$$EI(t) = PI(t)\frac{PAI(1h)_{CMHS}}{PSUL(1h)_{Calculated}} \tag{5.2}$$

gdje $EI(t)$ predstavlja procijenjenu direktnu dozračenost mjerenu u $\frac{\mathrm{W}}{\mathrm{m}^2}$, $PI(t)$je procijenjena razina vidljivosti sunca u postocima (predstavlja plavom linijom na slikama 5.4, 5.6, 5.8 i 5.10), $PAI(1h)_{MHS}$predstavlja iznos predviđene srednje direktne dozračenosti na temelju jednog sata koju određuje meteorološki servis, a izražava se u $\frac{\mathrm{W}}{\mathrm{m}^2}$, a $PSUL(1h)_{Calculated}$ predstavlja predviđenu srednju vrijednost predviđene vidljivosti sunca (mjeren u postocima koji može poprimiti vrijednost između 0 i 100%).

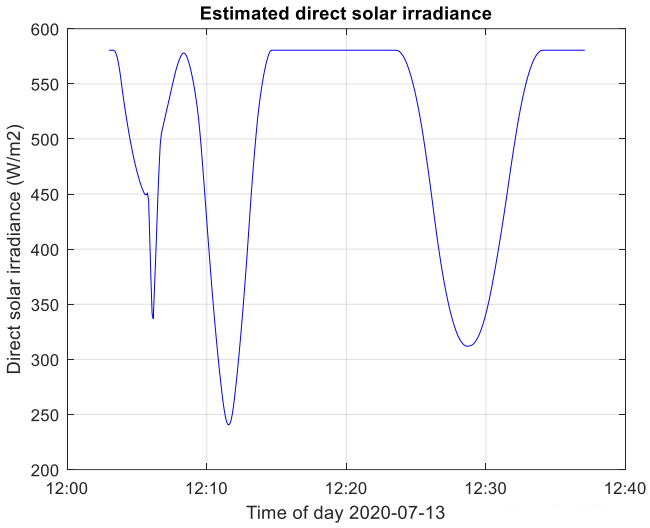
Glavna pretpostavka se temelji na tome da je prognoza meteorološkog servisa točna, a opisanom metodom određuje raspodjela zračenja unutar sat vremena. Standardna razina pogreške prikazana na slikama 5.5, 5.7, 5.9 i 5.11 pokazuje da je u rasponu od 10 do 20%. Rezultati pokazuju da je pogreška najveća u razdobljima kada oblaci započinju i prestaju prekrivati sunce, posebno ako su oblaci prozirni pa ih sunčeva svjetlost može probiti. Pogreška se određuje na način da se odredi odstupanje površine Sunca koje je zaklonjeno oblacima od površine kad Sunce nije zaklonjeno oblacima. U slučaju kad se površina Sunca počinje smanjivati, to znači da oblak počinje zaklanjati Sunce i to

smanjenje nije linearno. Kad započne zaklanjanje Sunca oblacima koji su prozračni, to rezultira povećanjem površine Sunca s obzirom na površinu Sunca kad nije zaklonjeno oblacima i to povećanje obrnuto proporcionalno utječe na razinu vidljivosti Sunca.

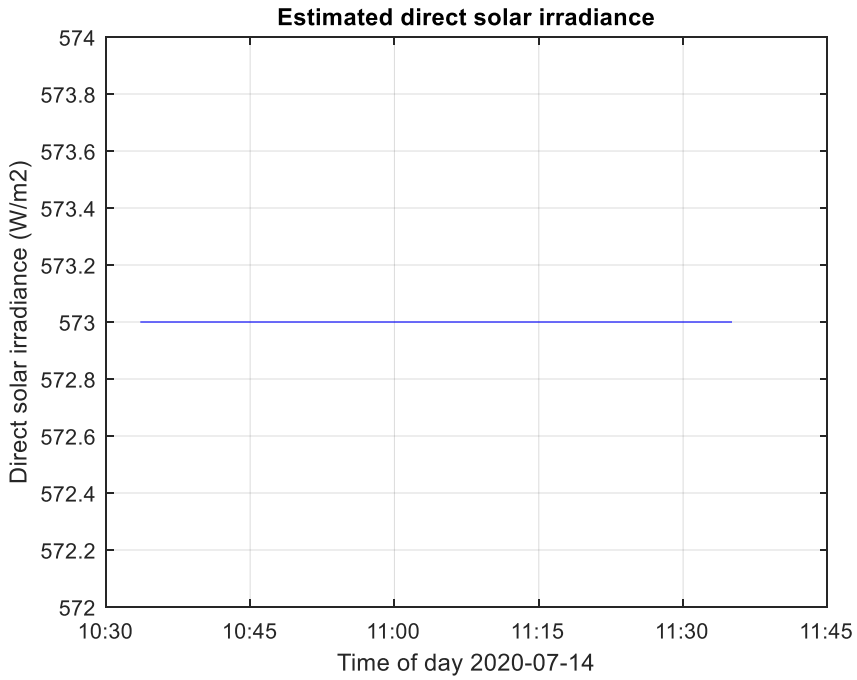
U slučaju kada je sunce smješteno blizu horizonta, s orijentacijom tamo gdje oblaci dolaze do horizonta (na primjer, istočna strana je na lijevom dijelu snimljenih slika neba) tijekom jutra (kao što je prikazano na slici 5.8), mogućnosti predviđanja su ograničene. To je zbog kratke udaljenosti između sunca i oblaka koji se tek pojave na horizontu neba i odmah se nalaze blizu pozicije sunca. Međutim, ukupna dozračenost obično je manja tijekom jutra pa je i apsolutna pogreška povezana s količinom energije niža. Stoga se optimalno predviđanje dobiva kada se sunce nalazi dalje od položaja oblaka (na primjer, usred dana, tijekom podneva). Meteorološki servis daje prognozirane vrijednosti direktnog sunčevog zračenja za odabrano područje za sljedeća 64 sata i daje prosječnu vrijednost sunčevog zračenja po satu. Na temelju predviđanja razine vidljivosti Sunca, može se procijeniti vrijednost direktnog sunčevog zračenja u ovisnosti o vremenu u zadanom intervalu, kao što je prikazano na slikama 5.12, 5.13, 5.14 i 5.15.



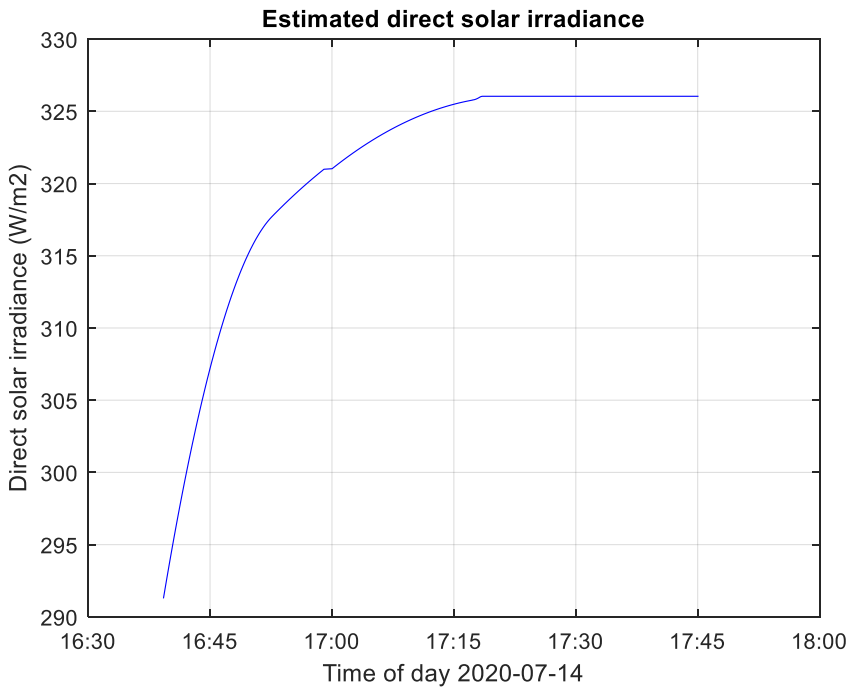
Slika 5.12. Predviđena razina direktne sunčeve dozračenosti na dan 13. 07. 2020. između 13:45h i 14:40h.



Slika 5.13. Predviđena razina direktne sunčeve dozračenosti na dan 13. 07. 2020. između 12:00h i 12:40h.



Slika 5.14. Predviđena razina direktne sunčeve dozračenosti na dan 14. 07. 2020. između 10:30h i 11:45h.



Slika 5.15. Predviđena razina direktne sunčeve dozračenosti na dan 13.07.2020. između 16:40h i 17:45h.

Budući da su za konture oblaka i Sunca aproksimirane oblikom pravokutnika s minimalnom površinom koja još uvijek opisuje konturu, algoritam se pokazao vrlo jednostavnim i zahtijeva prosječno vrijeme za procjenu pokrivenosti sunca između 5 i 6 sekundi. Simulacije su se izvodile na osobnom računalu s procesorom Intel Core i7-8750 (2,20 GHz) s 24 GB RAM-a (1 GB je korišteno za integrirano razvojno okruženje za Javu) s SSD diskovnim pogonom s uzastopnim čitanjem do 2800 MB/s.

6. Optimizacija tokova energije u mikromreži s fotonaponskim sustavom

Kod mikromreže s fotonaponskim sustavom potrebno je osigurati raspoloživost energije u traženom vremenu kao i mogućnost programiranog upravljanja snagom. Ukoliko obnovljivi izvor isporučuje energiju u elektroenergetski sustav i želi se da radi kao mikromreža, potrebno je unaprijed odrediti graničnu krivulju maksimalne proizvodnje i osigurati da se ta proizvodnja konstantno može ostvariti. Maksimalna proizvodnja je određena elektroenergetskim sustavom te garantira da će u svakom trenutku biti osigurana opskrba te količine električne energije. Temeljem satne prognoze srednje dozračenosti dobivene od meteorološkog servisa za koju se pretpostavlja da je potpuno točna moguće je, uzimajući u obzir korisnost pojedinog postrojenja zasnovanog na fotoelektričnom sustavu, definirati gornju graničnu krivulju proizvodnje za više dana unaprijed. Budući da prognoza dozračenosti daje srednju satnu dozračenost, sustav pohrane energije u mikromreži mora osigurati mogućnost isporuke energije jednake srednjoj vrijednosti unutar sata, bez obzira na fluktuacije snage uzrokovane prolazom oblaka. To znači da se u trenucima proizvodnje veće od srednje vrijednosti razlika proizvodnje i srednje vrijednosti pohranjuje, a u trenucima manje proizvodnje od srednje vrijednosti isporučuje iz raspoloživih pohrana.

Budući da se prognoza dozračenosti zasniva na satnoj srednjoj vrijednosti, a unutar sata se postupcima opisanim u prethodnim poglavljima pomoću kamere može dobiti preciznija predikcija dozračenosti sa sekundnom rezolucijom, sustav pohrane energije mora zadovoljiti potrebe pohrane unutar satnog intervala. Mikromreža u tom slučaju u jednu cjelinu integrira energetski izvor, sustav pohrane i uređaje za povezivanje na elektroenergetski sustav [39]. Uređaji za pohranu razmatrani u ovom radu su baterije, superkondenzatori i gorivne članke. Optimizacija tokova energije u mikromrežama temelji se na planiranju količine energije koja se pohranjuje ili se iz pohrane uzima u svrhu potrošnje unutar fiksno određenog vremenskog perioda kako bi se postigla maksimalna učinkovitost mikromreže [40], [41]. Superkondenzatori imaju veliku gustoću snage, baterije imaju veliku gustoću energije, dok sustav pohrane u vodiku ima velik kapacitet.

Baterijske pohrane se najčešće koriste u sustavima s mikromrežom [42]. Energija se pohranjuje u obliku naboja elektrokemijskih ćelija. Baterijski članci se mogu spajati u serijske i paralelne blokove te time omogućavaju postizane željenog kapaciteta i izlaznog napona. Principi rada baterija se temelje na materijalima kao što su olovo-kiselina, nikal-kadmij, natrij-sumpor i litij-ion. Svaki tip baterije ima svoje prednosti i nedostatke. Baterije temeljene na materijalima olovo-kiselina su jeftinije od ostali, dok natrij-sumpor baterije imaju gustoću energije od 151 KWh/m^3 [43]. Nikal-kadmij baterije imaju prednost po pitanju dužeg životnog ciklusa, a nikal-metal-hidrid baterije su prihvatljivije sa stanovišta zaštite okoliša. Litij-ionske baterije su skuplje od ostalih, međutim imaju najveću gustoću energije.

Pohrane energije temeljene na vodiku energiju pohranjuju pomoću metode elektrolizacije vode kako bi se proizveo vodik i kisik. Vodik se zatim dovodi do gorivne članke koja rekombiniraju vodik s kisikom te proizvode istosmjernu električnu struju. Takav sustav se još naziva i regenerativni gorivni članak (engl. *Regenerative Fuel Cell* - RFC) [42]. Glavne komponente ovog tipa pohrane energije su elektrolizator, stog gorivnih članaka, pohrana vodika te energetsko elektroničko sučelje za povezivanje ovog sustava s mikromrežom i elektroenergetskim sustavom. Prednost ovih pohrana energije je veliki kapacitet i manja emisija štetnih tvari. Osnovni nedostatak je niska učinkovitost (oko 50%) [44]. Budući da jedan gorivni članak može proizvesti napone manje od 1V, potrebno je koristiti stog ćelija kako bi se dobila željena razina izlaznog napona.

Kao pohrana energije koristite se i superkondenzatori koji pohranjuju energiju u elektrostatičkom obliku između ploča kondenzatora. Omogućavaju pohranu i pražnjenje velikih količina naboja u vrlo kratkom vremenu. Stoga su vrlo korisni u aplikacijama koje rade na višim frekvencijama i gdje su potrebne veće gustoće snage. Također ne zahtijevaju posebno složeno održavanje i imaju visoku učinkovitost [42].

U nastavku će biti opisan način dimenzioniranja sustava pohrane u ovisnosti o valnom obliku signala ulazne snage te amplitudi i frekvenciji promjene signala ulazne snage. S obzirom da se mikromreža temelji na fotonaponskom sustavu koji u znatnoj mjeri ovisi o dozračenoj sunčevoj energiji, tj. poziciji sunca i kretanju oblaka, optimalno dimenzioniranje sustava je vrlo bitno u poboljšanju karakteristika same mikromreže.

6.1. Dimenzioniranje sustava pohrane energije

Cilj dimenzioniranja sustava pohrane energije u mikromreži je da se odredi maksimalna količina energije za pohranu u satnom intervalu. Naime, s obzirom da je u petom poglavlju rada prikazan način predviđanja sunčeve dozračenosti te korištenje srednje procijenjene snage vrijednosti od meteorološkog servisa na razini jednog sata, taj period će biti razmatran i prilikom dimenzioniranja. Za dimenzioniranje sustava pohrane potrebno je razmotriti najgori slučaj, tj. slučaj proizvodnje kada je unutar sata potrebno pohraniti najviše energije. Količina pohranjene energije ovisi o amplitudi, frekvenciji i obliku fluktuacija oko srednje vrijednosti ulazne snage.

Ako je zadana srednja vrijednost generirane snage $\bar{P}$ koju predviđa meteorološki servis, stvarna snaga dobivena prema predviđenoj sunčevoj dozračenosti P, pri čemu je $t[h]$ satni period predviđanja. U tom slučaju izraz za pohranjenu energiju izgleda ovako:

$$E = \int_0^t (P - \bar{P})\, d\tau \tag{6.1}$$

Ukupni kapacitet pohrane određuje se prema maksimalnoj energiji koja može biti pohranjena u pohrane, a ovisi o valnom obliku signala snage, amplitudi promjene i frekvenciji promjene. Kako bi se izbjeglo da se proizvede manje energije od predviđene, na temelju predikcije potrebno je odrediti najgori slučaj u kojem bi došlo do najveće fluktuacije snage proizvodnje s obzirom na njenu srednju vrijednost koja se isporučuje u nadređeni sustav.

U slučaju pretpostavke da snaga generirana fotonaponskim modulom varira u sinusoidalnom obliku oko srednje vrijednosti, onda izraz za snagu $P(t)$ izgleda ovako:

$$P(t) = \bar{P} + A\sin(\omega t) \tag{6.2}$$

gdje je $P(t)$ trenutna snaga u vremenu t, $\bar{P}$ srednja predviđena razina snage temeljena na procjeni meteorološkog servisa, A razina amplitude predviđene snage, ω kutna frekvencija. Srednja vrijednost amplitude u obliku sinusoide u slučaju vremenskog trajanja koji je višekratnih perioda sinusoidalnog signala je nula. No, za sustav pohrane je važna vrijednost koja je veća od srednje vrijednosti. Dakle, pohranjena energija jednaka je integralu razlike proizvedene snage P(t) prema izrazu (6.2) i srednje vrijednosti $\bar{P}$ na

pozitivnom poluperiodu sinusoide. Iznos energije koji je potrebno pohraniti opisan je izrazom:

$$E = \int_0^{\frac{T_0}{2}} A \sin(\omega t)\, dt \tag{6.3}$$

gdje je *E* generirana energija u pozitivnom poluperiodu sinusoidnog signala, a $\frac{T_0}{2}$ vrijeme pozitivne poluperiode sinusoide. S obzirom da je:

$$T_0 = \frac{2\pi}{\omega} \tag{6.4}$$

te uzevši u obzir da za integral trigonometrijske funkcije vrijedi:

$$\int \sin cx\, dx = \frac{-1}{c} \cos cx \tag{6.5}$$

slijedi da je:

$$E = -A \frac{1}{\omega} \cos \omega t \tag{6.6}$$

Za interval [0, $\frac{T_0}{2}$] kod sinusoidnog oblika kad je prva polovica perioda signala u pozitivnom segmentu, izraz se pretvara u:

$$E = \frac{2A}{\omega} \tag{6.7}$$

Nakon što se izraz 6.4 uvrsti u izraz 6.7, dobiva se izraz:

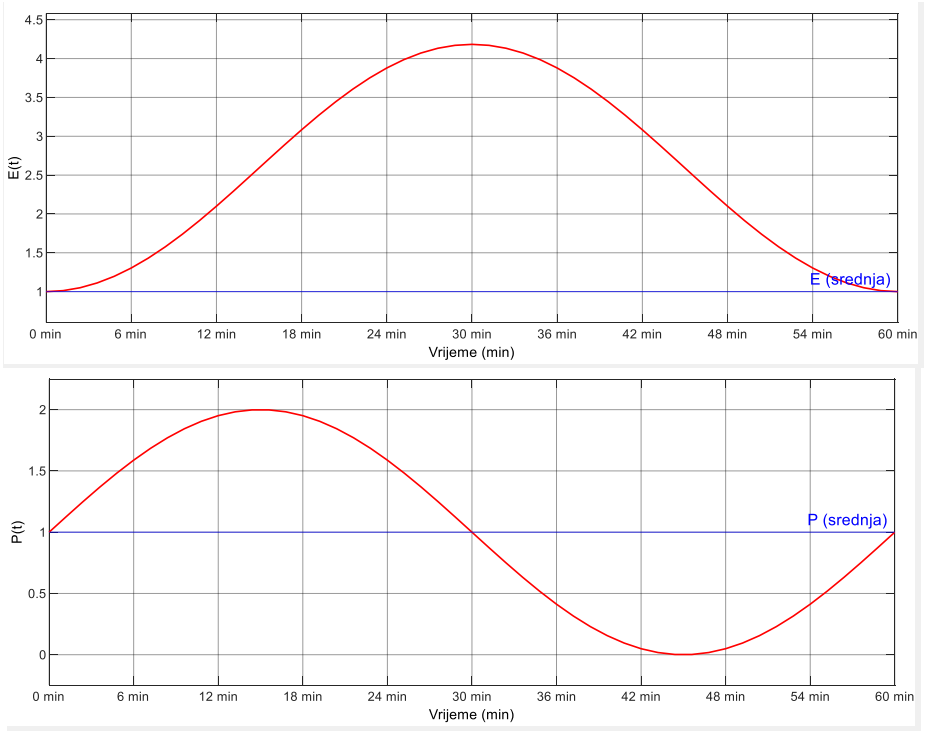
$$E = \frac{A\, T_0}{\pi} \tag{6.8}$$

Za postizanje maksimalne energije E_{max}, potrebno je postići maksimalnu moguću vrijednost za amplitudu *A* te minimalnu vrijednost za frekvenciju ω. Maksimalna moguća amplituda znači da je potrebno ostvariti maksimalna dozračenost, tj. Sunce nije prekriveno oblacima. Također, u tom slučaju generira se i maksimalna snaga P_{max}. Minimalna frekvencija predstavlja situacija kad postoji samo jedna promjena unutar perioda intervala od sat vremena T_h, što je predikcijski horizont DHMZ-a. To znači da je:

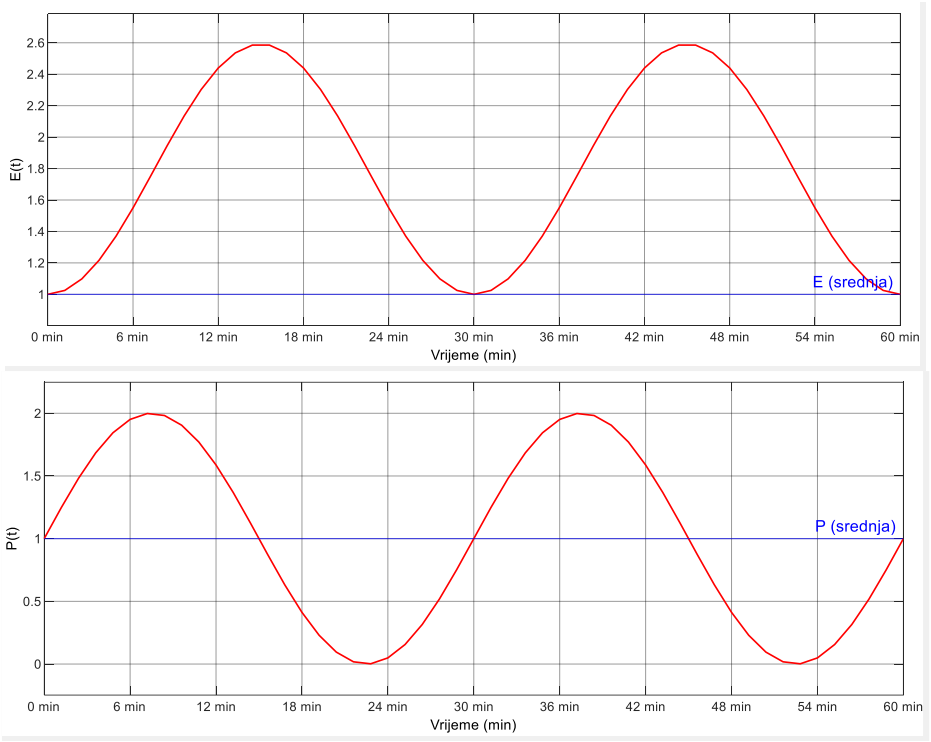
$$\omega_{min} = \frac{2\pi}{T_h} \tag{6.9}$$

6.1.1. Ovisnost krivulje iznosa energije o krivulji snage sinusoidalnog oblika

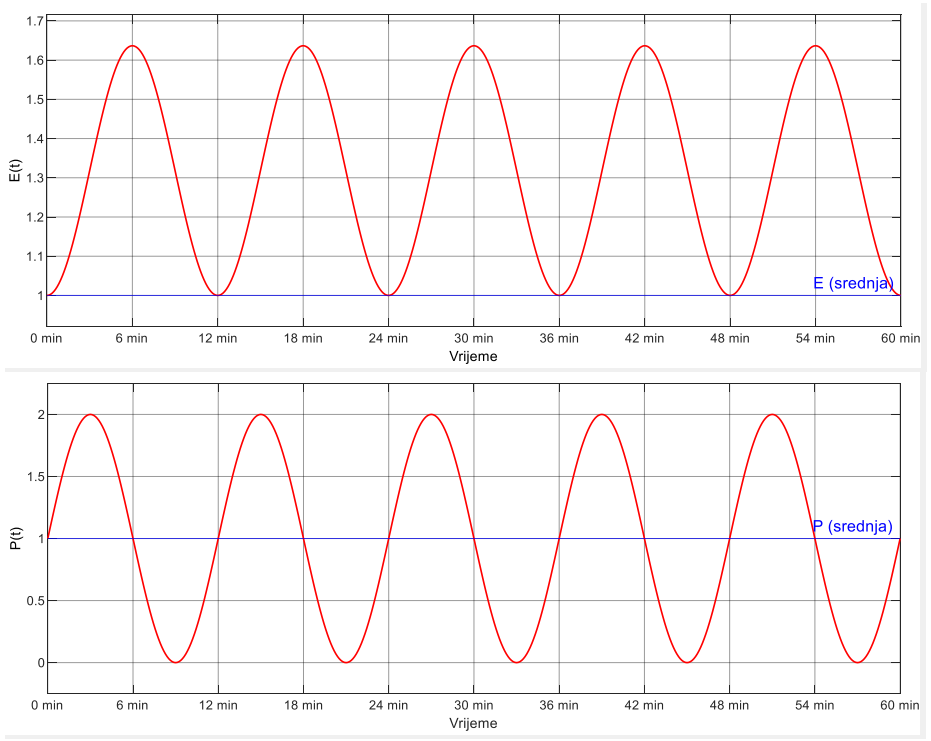
Sljedeće slike prikazuju na koji način se mijenja maksimalna energija u slučaju promjene amplitude snage, tj. promjene frekvencije. Na slici 6.1. prikazan je primjer krivulje energije u slučaju kad je amplituda A postavljena na 1, a frekvencija ima vrijednost $\frac{2\pi}{10}$. Srednja snaga koja se određuje od strane meteorološkog servisa je prikazana plavom crtom na slici. Ako se promijeni frekvencija na $\frac{2\pi}{5}$, a amplituda A ostane na 1, energija se mijenja kako je prikazano na slici 6.2. Dodatnim povećavanjem frekvencije na $\frac{2\pi}{2}$ uz istu amplitudu dobivaju se rezultati kao na slici 6.3., smanjivanjem na frekvenciju $\frac{2\pi}{15}$ rezultati izgledaju kao na slici 6.4., a s frekvencijom $\frac{2\pi}{20}$ rezultati su prikazani na slici 6.5.



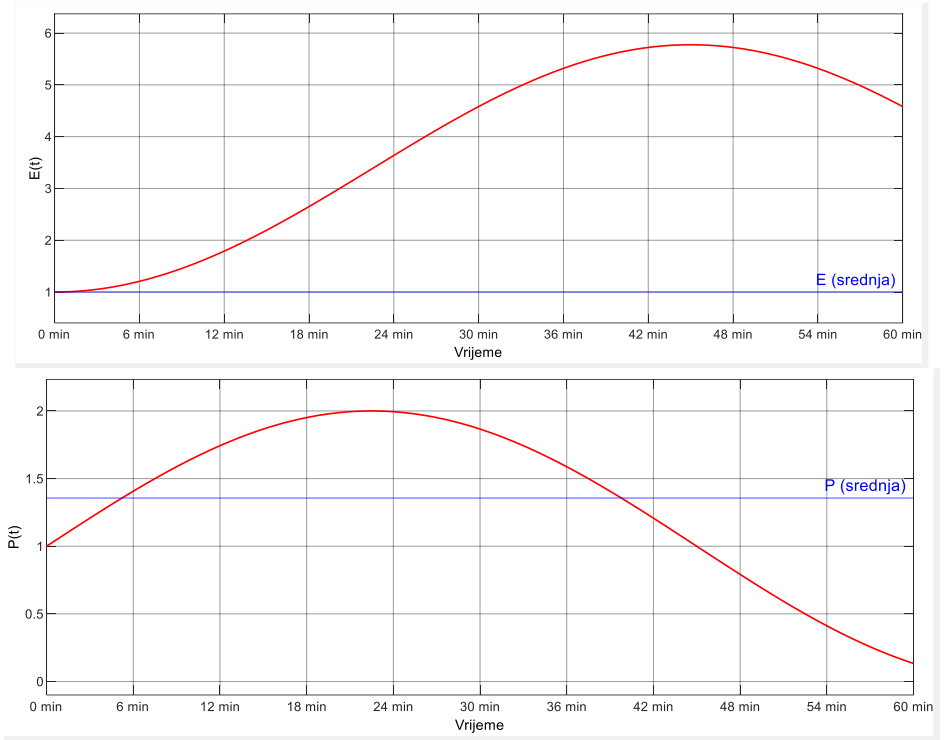
Slika 6.1. Primjer prikaza ovisnosti energije o sinusoidalnom signalu snage u slučaju amplitude A postavljene na vrijednost 1, a frekvencije na $\frac{2\pi}{10}$



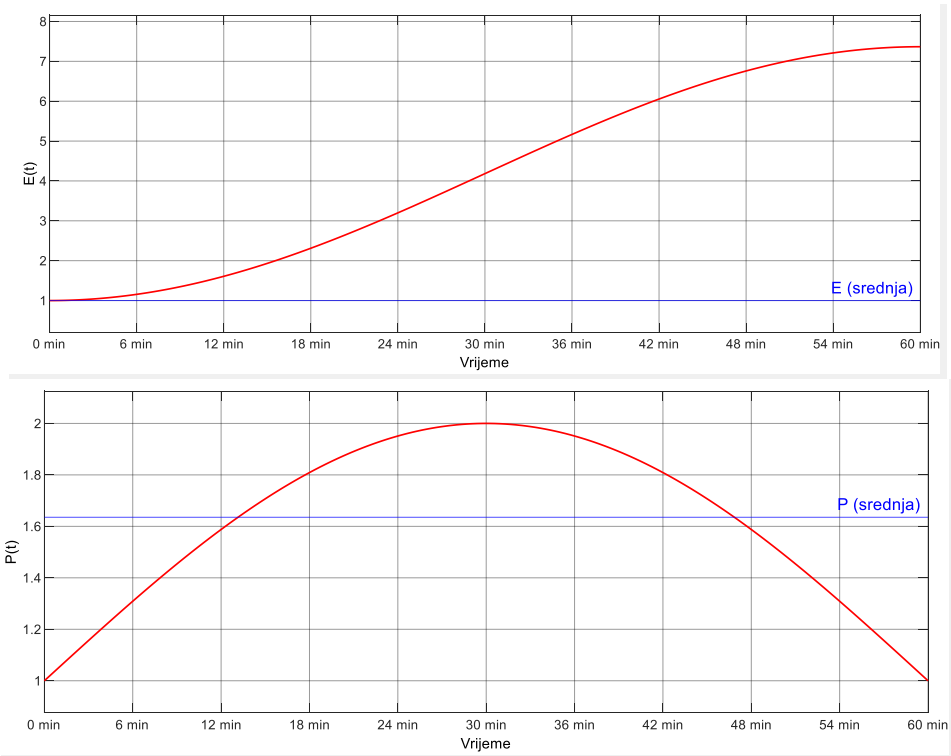
Slika 6.2. Primjer prikaza ovisnosti energije o sinusoidalnom signalu snage u slučaju amplitude A postavljene na vrijednost 1, a frekvencije na $\frac{2\pi}{5}$



Slika 6.3. Primjer prikaza ovisnosti energije o sinusoidalnom signalu snage u slučaju amplitude A postavljene na vrijednost 1, a frekvencije na $\frac{2\pi}{2}$



Slika 6.4. Primjer prikaza ovisnosti energije o sinusoidalnom signalu snage u slučaju amplitude A postavljene na vrijednost 1, a frekvencije na $\frac{2\pi}{15}$

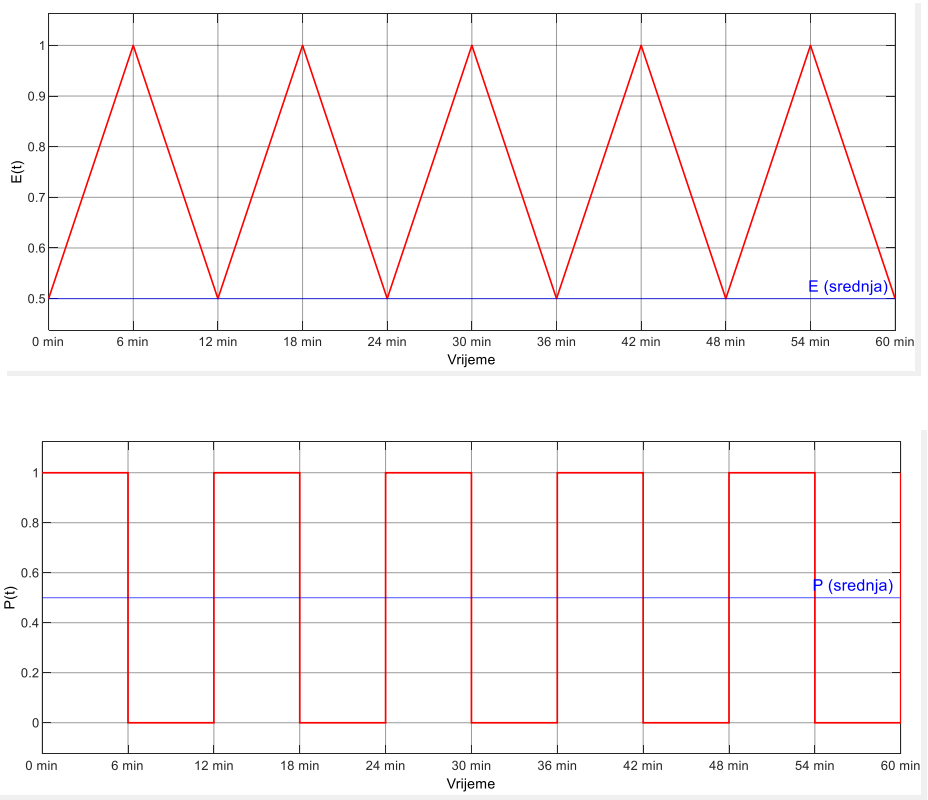


Slika 6.5. Primjer prikaza ovisnosti energije o sinusoidalnom signalu snage u slučaju amplitude A postavljene na vrijednost 1, a frekvencije na $\frac{2\pi}{20}$

Na slikama od 6.1. do 6.5., strmina krivulje snage *P* određuje brzinu porasta energija *E*.

6.1.2. Ovisnost krivulje iznosa energije o krivulji snage pravokutnog oblika

Osim sinusoidalnog valnog oblika, snaga može poprimiti i pravokutni oblik u slučaju većih oblaka na nebu koji naglo prekriju Sunce i potpuno ga prekrivaju svojom površinom. Na slici 6.6. prikazana je ovisnost energije o pravokutnom signalu snage u slučaju iznosa amplitude *A* postavljena na vrijednost 1.



Slika 6.6. Primjer prikaza ovisnosti energije o pravokutnom signalu snage u slučaju amplitude *A* postavljene na vrijednost 1

U slučaju pravokutnog napona, izraz za snagu bi izgledao ovako:

$$P(t) = \bar{P} + A[s(t) - s(t - \frac{T_0}{2})] \tag{6.10}$$

Pri čemu je $\bar{P}$ srednja vrijednost snage, A amplituda, a $s(t)$ predstavlja jediničnu skokovitu funkciju. U tom slučaju se energija određuje pomoću sljedećeg izraza:

$$E = \int_0^t (P - \bar{P})\, d\tau \tag{6.11}$$

$$E = \int_0^{\frac{T_0}{2}} A(s(t) - s(t - \frac{T_0}{2}))\, dt$$

Za interval [0, $\frac{T_0}{2}$] kod pravokutnog oblika kad je prva polovica periode signala u pozitivnom segmentu, izraz se pretvara u:

$$E = A\, \frac{T_0}{2} \tag{6.12}$$

Ako se izraz 6.12. usporedi s izrazom 6.8, može se zaključiti da se za pravokutni oblik snage, za istu frekvenciju i amplitudu, dobiva veća energija koju je potrebno pohraniti.

6.1.3. Ovisnost krivulje iznosa energije o popunjenosti

U prethodnim poglavljima je vidljivo da se u slučaju pravokutnog oblika krivulje snage dobiva maksimalna snaga, uz jednu promjenu amplitude unutar jednog periodu ($\omega = 1$) te maksimalnu amplitudu (puna snaga i potpuna dozračenost). Ako se koristi krivulja snage u obliku pravokutnog signala na početku perioda u trajanju od T_1 i maksimalne amplitude P_{max}, a u ostatku perioda ukupnog trajanja T_h, snaga je jednaka nuli. Slijedi da je srednja snaga $\bar{P}$ jednaka:

$$\bar{P} = \frac{1}{T_h} \int_0^{T_1} P_{max}\, d\tau = \frac{T_1}{T_h} P_{max} \tag{6.13}$$

Ako vrijedi da je:

$$T_1 = d\, T_h \tag{6.14}$$

pri čemu je $d \in [0, 1]$, tj. koeficijent povezan s trajanjem signala maksimalne snage P_{max}unutar cjelokupnog perioda T_h. Korištenjem izraza 6.13 i 6.14 dobiva se:

$$\bar{P} = \frac{d\, T_h}{T_h} P_{max} = d\, P_{max} \tag{6.15}$$

Maksimalna energija za pohranu u tom slučaju iznosi:

$$E_{max} = \int_0^{T_1} (P - \bar{P})\, d\tau \tag{6.16}$$

$$E_{max} = \int_0^{T_1} (P_{max} - d\ P_{max})\, d\tau$$

$$E_{max} = \int_0^{T_1} (1-d) P_{max}\, d\tau$$

$$E_{max} = T_1(1-d)P_{max} = dT_h(1-d)P_{max}$$

$$E_{max} = (-d^2 T_h + dT_h)P_{max}$$

Maksimalna energija za pohranu kao funkcija parametra popunjenosti d, prikazana je izrazom 6.16. Maksimalni iznos snage po popunjenosti d dobije se izjednačavanjem derivacije izraza 6.16 s nulom.

$$\frac{d}{\mathrm{d}d} E_{max} = (-2dT_h + T_h)P_{max} = 0 \tag{6.17}$$

Dijeljenjem izraza 6.17 s P_{max} dobiva se izraz:

$$\begin{gathered} (-2dT_h + T_h) = 0 \\ (2dT_h) = T_h \\ d = \frac{1}{2} \end{gathered} \tag{6.18}$$

Prema tome se može zaključiti da je najgori slučaj za pohranu, tj. slučaj kad je potrebno pohraniti najviše energije u satnom periodu kad signal maksimalne snage traje polovicu ukupnog perioda T_h. Uvrštavanjem izraza 6.18 u izraz 6.16. dobiva se:

$$\begin{gathered} E_{max} = (-0{,}25T_h + 0{,}5T_h)P_{max} \\ E_{max} = 0{,}25T_h P_{max} \end{gathered} \tag{6.19}$$

Prema izrazu 6.19 može se zaključiti da je maksimalna potrebna pohranjena energija određena s $\frac{1}{4}$ maksimalne satne proizvodnje.

Kombiniranjem izraza 6.15 i 6.16 s izrazom maksimalne energije koju je potrebno pohraniti (6.19), maksimalna pohranjena energija može se izraziti za najgori slučaj pomoću srednje i maksimalne snage unutar satnog perioda:

$$E_{max} = T_h P_{max}(-d^2 + d)P_{max}$$

$$E_{max} = T_h P_{max} d(1-d)$$

$$E_{max} = T_h P_{max} \frac{\bar{P}}{P_{max}} \left(1 - \frac{\bar{P}}{P_{max}}\right) \tag{6.20}$$

$$E_{max} = T_h \bar{P} \left(\frac{P_{max} - \bar{P}}{P_{max}}\right)$$

$$E_{max} = T_h P_{max} \frac{\bar{P}}{P_{max}} (P_{max} - \bar{P})$$

Za određivanje ukupne učinkovitosti sustava gdje je P proizvedena snaga, $\bar{P}$ srednja snaga, P_p ukupna pohranjena snaga, $E_p(T_h)$ ukupna pohranjena energija, a $E_p(0)$ pohranjena energija na početku, može se napisati:

$$\eta = \frac{\int_0^{T_h} \left(P - P_p\right) d\tau + E_p(T_h) - E_p(0)}{\int_0^{T_h} P \ d\tau}$$

$$\eta = \frac{\int_0^{T_h} P \, d\tau - \int_0^{T_h} P_p \, d\tau + E_p(T_h) - E_p(0)}{\int_0^{T_h} P \ d\tau} \tag{6.21}$$

$$\eta = 1 - \frac{\int_0^{T_h} P_p \, d\tau + E_p(T_h) - E_p(0)}{\int_0^{T_h} P \ d\tau}$$

6.2. Karakteristike korištenih pohrana energije

Pohrane energije u mikromrežama predstavljaju međuspremnik u koji se može pohranjivati energija za kasnije korištenje ili kao resurs s kojim se smanjuje neravnoteža

između potraživane i proizvedene energije. U slučaju koji se razmatra u okviru ove disertacije koriste se superkondenzator, baterija i vodik.

Struktura superkondenzatora se može promatrati kao serijski spoj dva dvoslojna kondenzatora s elektrostatičkom kapacitetom (engl. *Electric Double Layer Capacitor* - EDLC), a ne kao struktura s konvencionalnim krutim dielektrikom koji koriste obični kondenzatori [43]. Superkondenzator se još naziva i ultrakondenzator. Bez korištenja kemijskih procesa, električna energija se izravno pohranjuje uz kratke vremenske odzive. Kapacitet i gustoća energije takvih uređaja je za tri veličine veća od elektrolitskih kondenzatora. Iznosi kapaciteta se kreću od 5 F do 2700 F.

Baterije pohranjuju energiju u elektrokemijskom obliku i mogu imati različite kapacitete i učinkovitost. Osnovni tipovi baterija koje se koriste kao pohrane energije u mikromrežama su olovne baterije, nikal-željezne nikal-kadmijske, nikal-metal-hidridne i litij-ionske baterije. Olovne baterije postoje najdulje na tržištu i spadaju u najjeftinije pohrane, ali imaju ograničen broj ciklusa punjenja i pražnjenja te predstavljaju neprihvatljivo i neisplativo rješenje u sustavu mikromreže. Nikal-kadmij baterije su po tom pitanju bolje, imaju veću gustoću energije, duži životni vijek i niže troškove održavanja. Nikal-metal-hidridne baterije su prihvatljivije za okoliš te imaju veći kapacitet od olovnih baterija. Litij-ionske baterije imaju najveću gustoću energije, ali su i najskuplje [43]. Prema navedenom, za korištenje u mikromrežama najprikladnije su nikal-metal-hidridne baterije.

Vodik omogućava pohranu energije na ekološki prihvatljiv način, jer kod sagorijevanja ispušta samo vodenu paru u okoliš [45]. Gorivne ćelije vodika koriste čisti vodik te ga na temelju elektrokemijske reakcije pretvaraju u električnu energiju, toplinsku energiju i vodu [46]. Kako bi se povećao kapacitet, koristi se niz ćelija grupiranih u zajedničko polje.

U tablici 6.1. [42]–[44] prikazane su karakteristike korištenih pohrana u modelu opisanom u sljedećem poglavlju. U modelu su se koristili podaci o učinkovitosti i vremenu odziva te kapacitetu.

Tablica 6.1. Usporedba korištenih pohrana energije

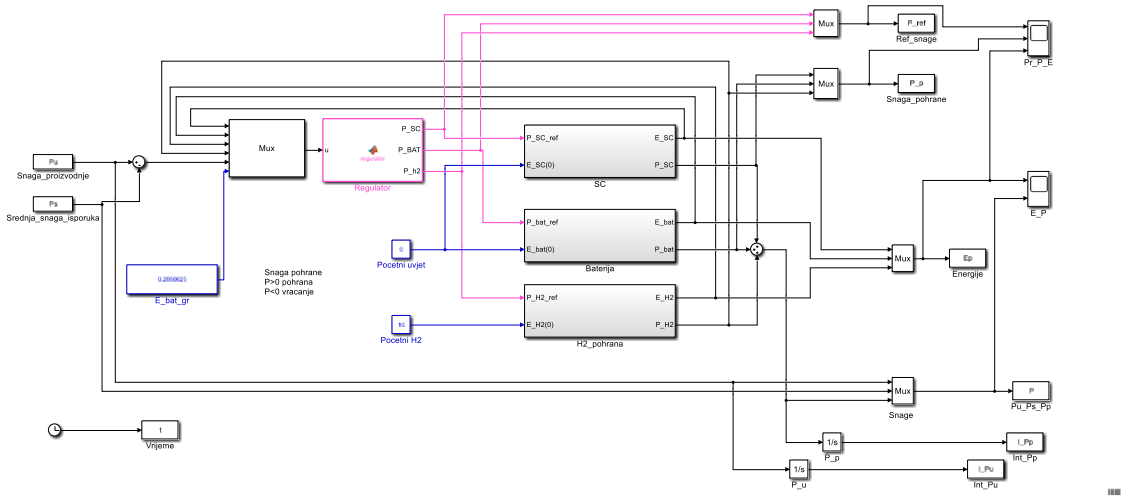
Tip pohrane	Korisnost (%)	Gustoća energije (Wh/kg)	Gustoća snage (W/kg)	Vrijeme odziva (ms)
Superkondenzator	95	<50	4000	5
Baterija	60-90	20-200	25-1000	30
Vodik	50	33000	33600	60000

U sljedećem poglavlju biti će opisan način optimiranja tokova energije korištenjem modela u Simulinku, temeljenog na opisane tri vrste pohrane energije.

6.3. Optimiranje tokova energije

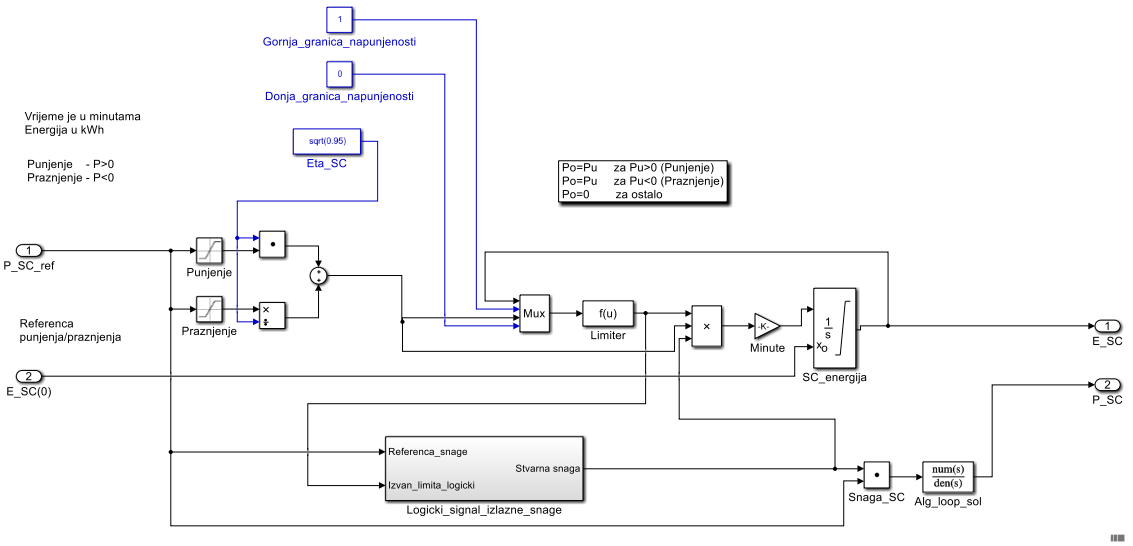
Kako bi se postiglo optimalno upravljanje tokovima energije u mikromreži s tri različite pohrane energije (superkondenzator, baterija i vodik), u MATLAB/Simulink-u je razvijen model kojim su obavljeni simulacijski eksperimenti optimiranja. Svaka od pohrane energije ima svoj kapacitet, trenutačnu napunjenost, učinkovitost i brzinu odziva, čime je potrebno optimalno upravljati kako bi se postigla pohrana maksimalne količine energije koja osigurava stabilan rad sustava.

U petom poglavlju je opisano na koji način se analizom uzastopnih slika neba predviđa razina dozračenosti u sljedećih sat vremena. Ta predikcija se koristi kao ulazni podatak MATLAB/Simulink modela koji upravlja tokovima energije, vodeći računa o napunjenosti svake od pohrana te usmjeravajući tok energije u onu pohranu u koju je moguće pohraniti dodatnu energiju (nije napunjen do kraja). Nakon analize 12 uzastopnih slika može se odrediti predikcija dozračenosti (kako se svaka slika neba kreira nakon 5 sekundi, analiza ukupno traje 12 sekundi) i ponavljati se svake minute kako bi se predviđanje dodatno korigiralo ako se promijenila situacija s oblacima po pitanju zasjenjenja sunca.



Slika 6.7. MATLAB/Simulink model za optimiranje tokova energije

Na slici 6.7. prikazan je opisani MATLAB/Simulink model. Sastoji se od blokova „SC“ koji predstavlja superkondenzator kao pohrana energije (detaljnija struktura bloka je prikazana na slici 6.8), „Baterija“ koji predstavlja bateriju te „H2_pohrana“ koji predstavlja vodik kao treći tip pohrane energije. Blok „Regular“ služi za upravljanje pohranama energije po pitanju pohrane energije u njih ili trošenja energije iz njih. Na izlazu je blok sumator koji računa ukupnu snagu pohrane ili pražnjenja, pri čemu svaka pohrana ima svoj status napunjenosti.

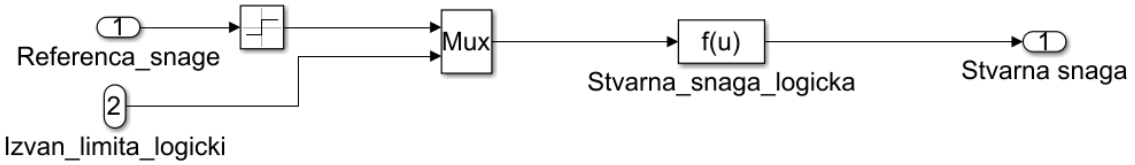


Slika 6.8. MATLAB/Simulink model za blok koji predstavlja superkondenzator

Na slici 6.8. ulazni parametar označen s „P_SC_ref“ predstavlja referentnu snagu koja definira količinu snage koju regulator s modela prikazanog na slici 6.7. pohranjuje ili dohvaća iz superkondenzatora. Ako se radi o punjenu superkondenzatora, radi se o pozitivnom iznosu snage, a u slučaju pražnjenja se radi o negativnom iznosu snage. U cilju postizanja tog ponašanja u model su dodana dva limitera: „Punjenje“ postavljanja nižu granicu na „0“, a „Praznjenje“ postavlja višu granicu na „0“. Kod punjena se snaga množi s učinkovitošću superkondenzatora označenog s „Eta_SC“ (manji dio energije se pohranjuje), a kod pražnjenja se dijeli s učinkovitošću superkondenzatora (jer se povlači više snage iz pohrane) kako bi se aplicirao efekt koeficijenta učinkovitosti. Nakon toga se u modelu ograničavaju gornja i donja granica napunjenosti korištenjem komponenata „Gornja_granica_napunjenosti“ i „Donja_granica_napunjenosti“. U slučaju da kad se pohrana superkondenzatora napuni do kraja, u njega se više ne pohranjuje dodatna energija.

Pražnjenje se postiže tako da se na ulaznu komponentu „P_SC_ref“ dovodi negativan iznos, ali samo do trenutna kad se pohrana superkondenzatora isprazni do iznosa snage „0“. To se postiže komponentom „Limiter“ koja provjerava je li stanje energije na integratoru „SC_energija“ između donje i gornje granice napunjenosti. Ulazna energija u kWh se pomoću komponente „Minute“ dijeli za 60, kako bi se vremenska jedinica pretvorila u sate. Početno stanje napunjenosti se dovodi na ulaznu komponentu „E_SC(0)“ koja se dovodi na integrator „SC_energija“. Prvi izlaz iz modela je „E_SC“ i predstavlja status energije.

Za određivanje izlazne snage koristi se blok „Logicki_signal_izlazne_snage“. Ulaz je ulazna snaga koja se dovodi na ulaz „Referenca_snage“ te se provjerava da li je iznos između donje i gornje granice napunjenosti (što se dovodi na ulaz „Izvan_limita_logicki“). Izgled bloka „Logicki_signal_izlazne_snage“ prikazan je na slici 6.9.



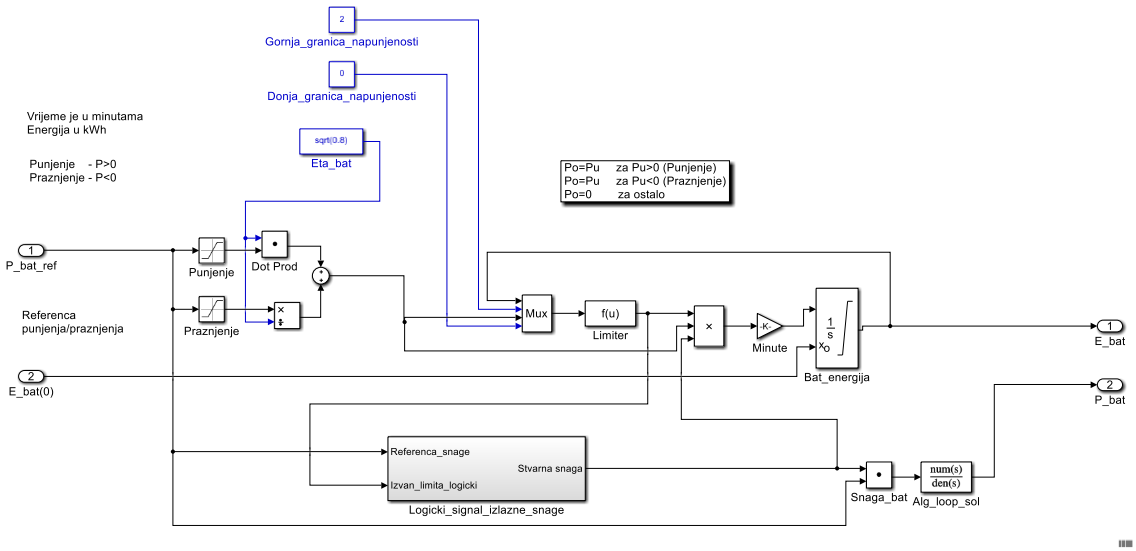
Slika 6.9. Izgled bloka „Logicki_signal_izlazne_snage“

Na slici 6.9. prikazan je ulazni signal „Referenca_snage“ koji se prosljeđuje na „Sign“ funkciju koja vraća vrijednosti iz domene (1, 0, -1). Ulazni signal „Izvan_limita_logicki“ definira stanje u kojem je vrijednost snage između definirane gornje i donje granice te omogućava očekivanu funkcionalnost punjenja i pražnjenja (odnosno pohrana energije nije potpuno pun ili prazan). Element koji predstavlja funkciju „Stvarna_snaga_logicka“ provjerava je li u toku punjenje ili pražnjenje. Ako pohrana nije potpuno pun ili prazan, na izlaz vraća logičku vrijednost „1“, što rezultira preslikavanjem ulazne snage na izlaz u modelu na slici 6.8. U ostalim slučajevima na izlaz se vraća „0“. To je slučaj kad na početku simulacije započne punjenje superkondenzatorske pohrane i kad se napuni do kraja, punjenje prestaje, ali ulazna snaga se i dalje troši. Kad počinje faza pražnjenja, istog trenutka se kreće s pražnjenjem, a brzina pražnjenja ovisi o koeficijentima učinkovitosti. Kad pohranjena energija dođe do „0“, faza pražnjenja prestaje i izlazna energija pada na „0“ bez obzira na referentnu vrijednost. Prije preslikavanja ulazne snage na izlaz postoji regulacijski PT_1 član (označen s „Alg_loop_sol“) koji u direktnoj grani ima integrator s pojačanjem 0.1 kojim se sprečavaju algebarske petlje.

U slučaju kad komponenta „Logicki_signal_izlazne_snage“ vraća „1“, ulazna snaga se prenosi i množi s tim koeficijentom „1“ te je time predstavljena razina izlazne snage na signalu „P_SC“ prikazanom na slici 6.8. Budući da se ulazna snaga kod pohrane množi s koeficijentom učinkovitosti, u konačnici se pohranjuje manje energije od ulazne energije. Količina snage koja se troši je definirana referentnom razinom.

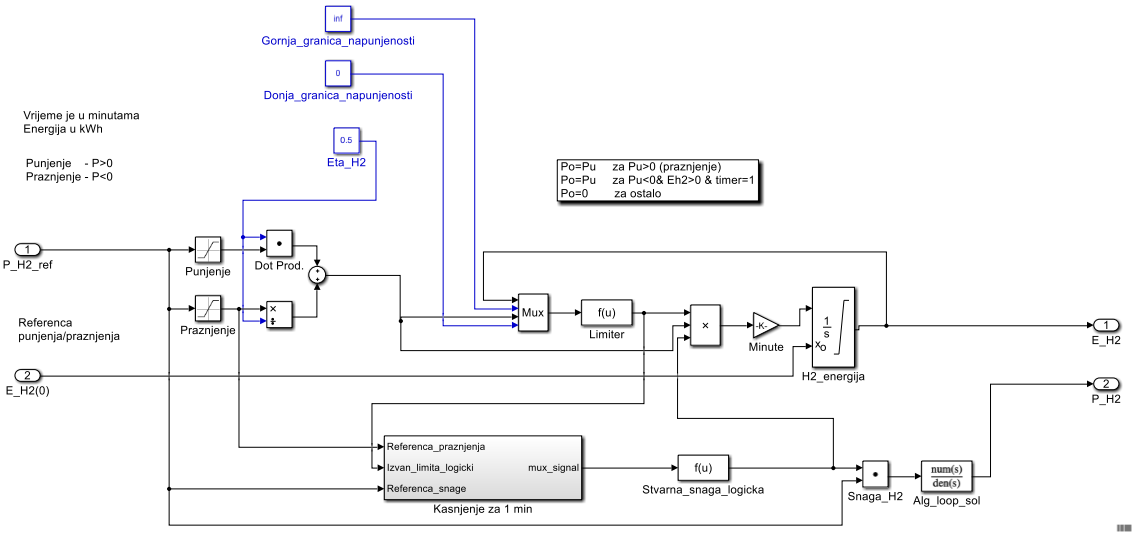
Na sličan način funkcionira i modul koji na slici 6.7. predstavlja bateriju, što je prikazano na slici 6.10., samo što ima postavljene drugačije koeficijente učinkovitosti.

Što se tiče pohrane u vodik, u tu pohranu je moguće pohraniti veliku količinu energije pa se njegova gornja granica napunjenosti može postaviti na beskonačnu veličinu (označeno s „inf“ na slici 6.11). Donja granica napunjenosti je postavljena na vrijednost „0“. Limiter funkcionira na isti način kao i kod superkondenzatora i baterije. S obzirom na karakteristike vodika kao pohrane energije, njegovo vrijeme odziva je znatno duže, što je prikazano u Tablici 1.



Slika 6.10. Simulink model za blok koji predstavlja bateriju

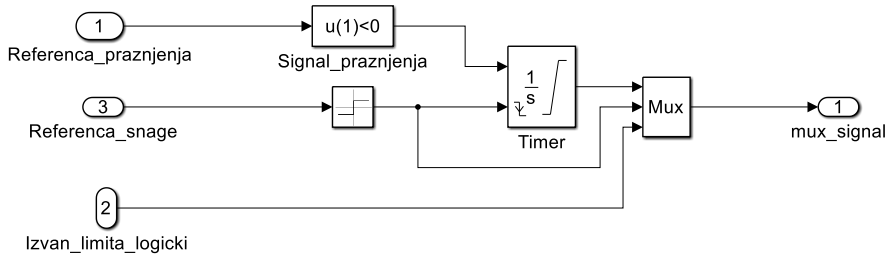
To znači da se prebacivanja u stanje pražnjenja prebacuje sporije, s jednom minutom zakašnjenja. Radi toga logički signal izlazne snage, označen s „Kasnjenje za 1 min“ na slici 6.11. funkcionira drugačije nego u slučaju superkondenzatora i baterije te taj modul prima tri ulazna parametra . Blok „Kasnjenje za 1 min“ prikazan je na slici 6.12.



Slika 6.11. Simulink model za blok koji predstavlja vodik

Ulazni signal „Referenca_praznjenja“ na slici 6.12. dovodi se na blok označen sa „Signal_praznjenja“, koji vraća „1“ na ulazu ako se radi o fazi pražnjenja i dovodi taj signal na integrator označen s oznakom „Timer“ koji simulira kašnjenje od jedne minute. Ta vrijednost je uvijek predstavljen samo negativnim dijelom snage, s obzirom da je taj

signal doveden nakon bloka „Praznjenje“ (na slici 6.11.) kojem je gornja granica postavljena na 0. Referenca snage (ulaz označen brojem „3“) koristi „Signum“ funkciju koja pretvara referentni iznos snage u vrijednost „1“ (u slučaju pozitivne snage i punjenja) ili „0“ u slučaju negativne snage i pražnjenja.



Slika 6.12. Izgled bloka „Kasnjenje za 1 min“

Integrator „Timer“ ima konfiguriranu postavku „Externalreset“ na „*falling*“, što označava padajući brid, tj. opisuje situaciju prelaska iz načina rada punjenja ili iz stacionarnog stanja (kad se ne puni) u način rada pražnjenja. Takav padajući brid postavi integrator „Timer“ u stanje „0“, nakon čega integrator počne rasti do vrijednosti „1“ (što je postavljeno kao „*Upper saturation limit*“). Na kraju se sve tri vrijednosti, izlaz iz „Timer“ integratora koji vraća „1“ nakon isteka jedne minute (opisano kašnjenje), izlaz iz „Signum“ funkcije, kao i logički izraz koji predstavlja da je snaga i dalje izvan limita (pohrana nije prazna ni puna), dovode na blok multipleksora „Mux“ i vraćaju kao izlazna vrijednost u glavni model prikazan na slici 6.11. Ta izlazna snaga se dovodi na funkciju „Stvarna_snaga_logicka“ koja provjerava je li referenca snage pozitivna (u slučaju punjenja) ili je referenca snage negativna (u slučaju pražnjenja). Integrator (označen s „Time“ na slici 6.12) na izlaz vraća „1“ (što znači da je istekla jedna minuta) i još uvijek pohrana nije prazna (tj. nalazi se u statusu „Izvan limita“). Ako je jedan od ta dva uvjeta ispunjen, funkcija vraća „1“ i onda se snaga prenosi na izlaz kroz množitelj „Snaga_H2“.

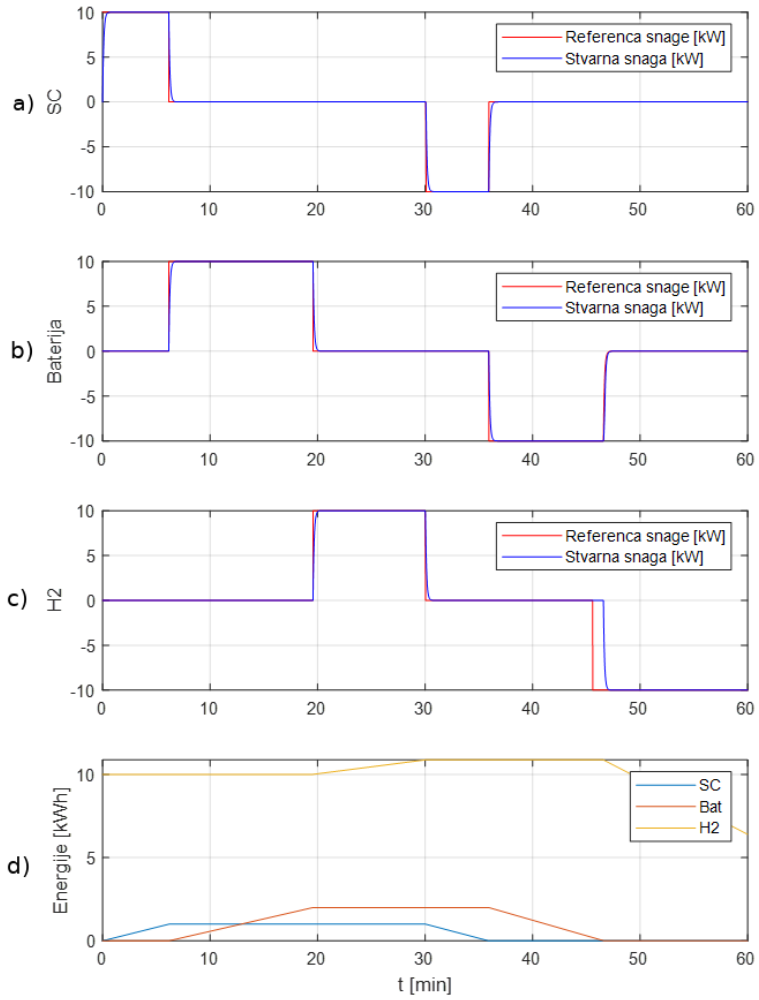
Cilj optimiranja je odrediti iznos energije „E_bat_gr“ (u modelu prikazanom na slici 6.8) koja mora biti pohranjena unutar baterije kako bi se osiguralo da se vodik može dovesti do gorivne ćelije i pokrenuti rad gorivne ćelije (unutar jedne minute). Energija iz baterije osigurava kontinuiranu isporuku energije te eliminira efekt sporijeg odziva vodika i njegovo prebacivanje u stanje pražnjenja. U prilozima D i E prikazana je funkcija cilja koja pronalazi granični iznos „E_bat_hr“ te primjer izvođenja ciklusa optimiranja u MATLAB-u.

6.4. Eksperimentalni rezultati

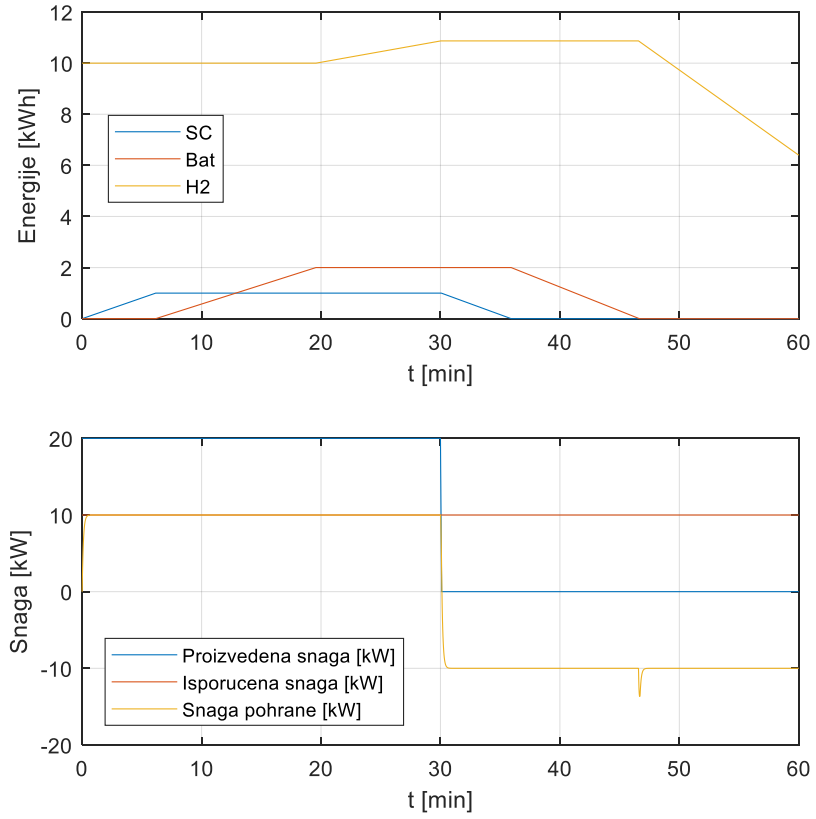
U nastavku su prikazani rezultati koji pokazuju snage i energije superkondenzatora, baterije i vodika tijekom punjenja i pražnjenja. Na slici 6.13 su prikazane dobivene vrijednosti u slučaju kad je proizvedena snaga u prvih pola sata postavljena na 20 kW, a preostalih pola sata 10 kW (označeno plavom krivuljom na slici 6.14). Narančasta krivulja na slici 6.14. označava isporučenu snagu, a žuta krivulja označava snagu pohrane. Pozitivni iznos snage označava fazu punjenja, a negativni označava fazu pražnjenja. Dio žute krivulje između 40. i 50. minute na grafu koji naglo mijenja vrijednost (u obliku „šiljka“) predstavlja trenutak, tj. fazu pokretanja vodika koja traje jednu minutu nakon čega može početi pražnjenje pohrane vodika. Taj dio krivulje oblikovan je na taj način zbog dodavanja člana za kompenziranje algebarske petlje.

Vidljivo je da prvo započinje faza punjenja superkondenzatora (plava krivulja na slici 6.13 d)), nakon čega slijedi punjenje baterije (narančasta krivulja na slici 6.13, graf označen s „Energije [kWh]“). Nakon što se napuni i baterija slijedi punjenje vodika (žuta krivulja na slici 6.13 d)). Početna energija pohranjena u vodik je 10 kWh pa vrijednost energije pohranjene u vodik na početku raste sve do 30. minute. Proizvedena snaga u prvih pola sata ima vrijednost 20 kW. Nakon što proizvedena snaga padne na nulu (što traje sljedećih 30 minuta), kreće pražnjenje vodika, a zatim baterije. Kod baterije je bitno primijetiti da se dosezanjem granične vrijednosti dobivene optimiranjem pokreće zagrijavanje vodika kako bi on mogao početi s pražnjenjem kad se baterija isprazni.

Na grafovima „SC“, „Baterija“ i „H2“ je vidljiva ovisnost referentne snage s obzirom na stvarni iznos snage, pri čemu pozitivna snaga označava pohranjivanje energije, a negativna pražnjenje. Nakon što se napuni superkondenzator na iznos od 1 kW, kreće punjenje baterija do iznosa 2 kW, a nakon toga se preostala energija pohranjuje u vodik. Za navedene iznose energije se optimiranjem dobiva granična vrijednost energije u bateriji 0,2050kWh.



Slika 6.13. Ovisnosti energija superkondenzatora („SC“), baterije i vodika („H2“) o snagama punjenja i pražnjenja u periodu od jednog sata (SC ima snagu 1 kW, baterija 2 kW, a početna napunjenost vodika je 10 kWh) za snagu proizvodnje postavljenu na 50% maksimalne snage

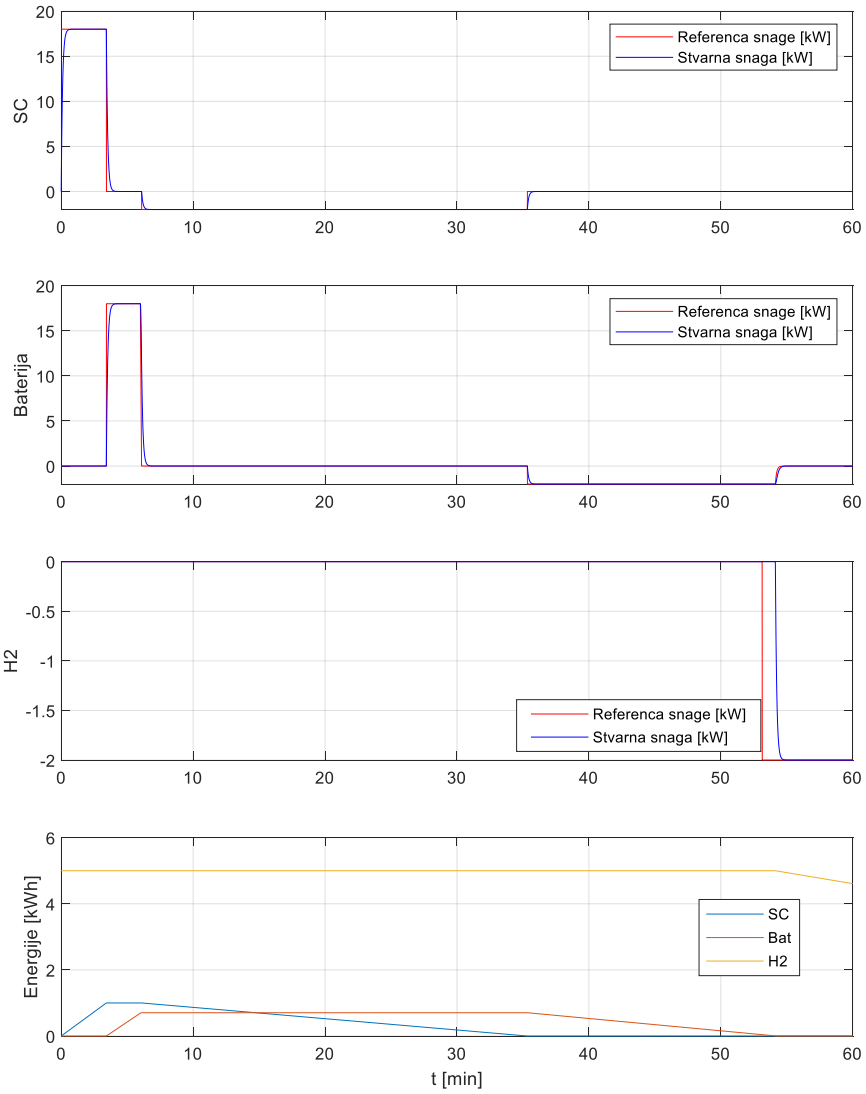


Slika 6.14. Ovisnost energija pohrane o snazi. SC ima snagu 1 kW, baterija 2 kW, a početna napunjenost vodika je 10 kWh. Snaga proizvodnje postavljena je na 50% maksimalne snage

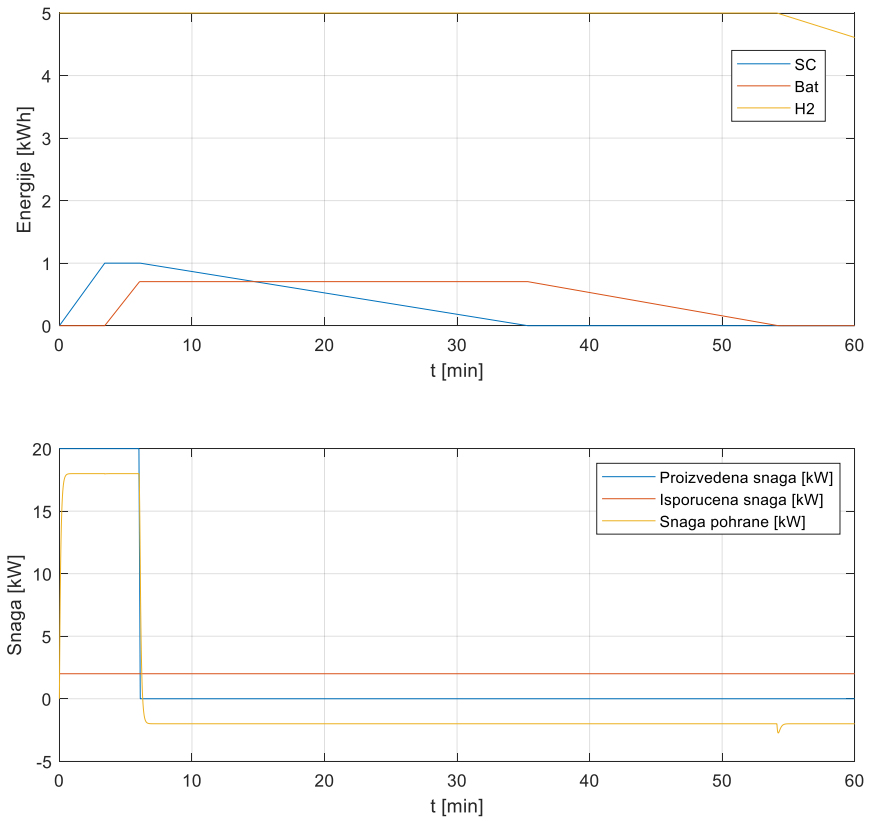
Ukupna učinkovitost sustava η_{uk} ovisi o učinkovitostima svih pohrana energije u sustavu. Temelji se na omjeru količine pohranjene energije i količine energije koja se isprazni iz pohrane. U slučaju kad se u sustav pohranjuje maksimalna količina energije, najveći dio energije se pohrani u pohranu vodika s najmanjom učinkovitošću što rezultira s najmanjoj ukupnom učinkovitošću sustava. Za navedene parametre na slikama 6.13 i 6.14 simulacijom se dobiva ukupna učinkovitost sustava u iznosu od $\eta_{uk} = 0{,}6399$.

Na slikama 6.15 i 6.16 opisan je slučaj kad je snaga fotonapona postavljena na 10% maksimalne snage (6 minuta unutar sat vremena je snaga postavljena na 20 kW, a ostale 54 minute na 0 kW), dok je srednja isporučena snaga 2 kW. Početna napunjenost vodika je postavljena na 5 kWh. Vidljivo je da se superkondenzatora puni u potpunosti, baterija samo djelomično te ne preostaje snage za punjenje vodika. Pražnjenje se odvija sporije zbog manje isporučene srednje snage nego na slikama 6.13 i 6.14. Nakon što se isprazne superkondenzator i baterije, započinje pražnjenje vodika. Granična vrijednost energije

baterije nakon optimiranja iznosi 0,041 kWh. Ukupna učinkovitost u ovom slučaju iznosi $\eta_{uk} = 0{,}9218$.

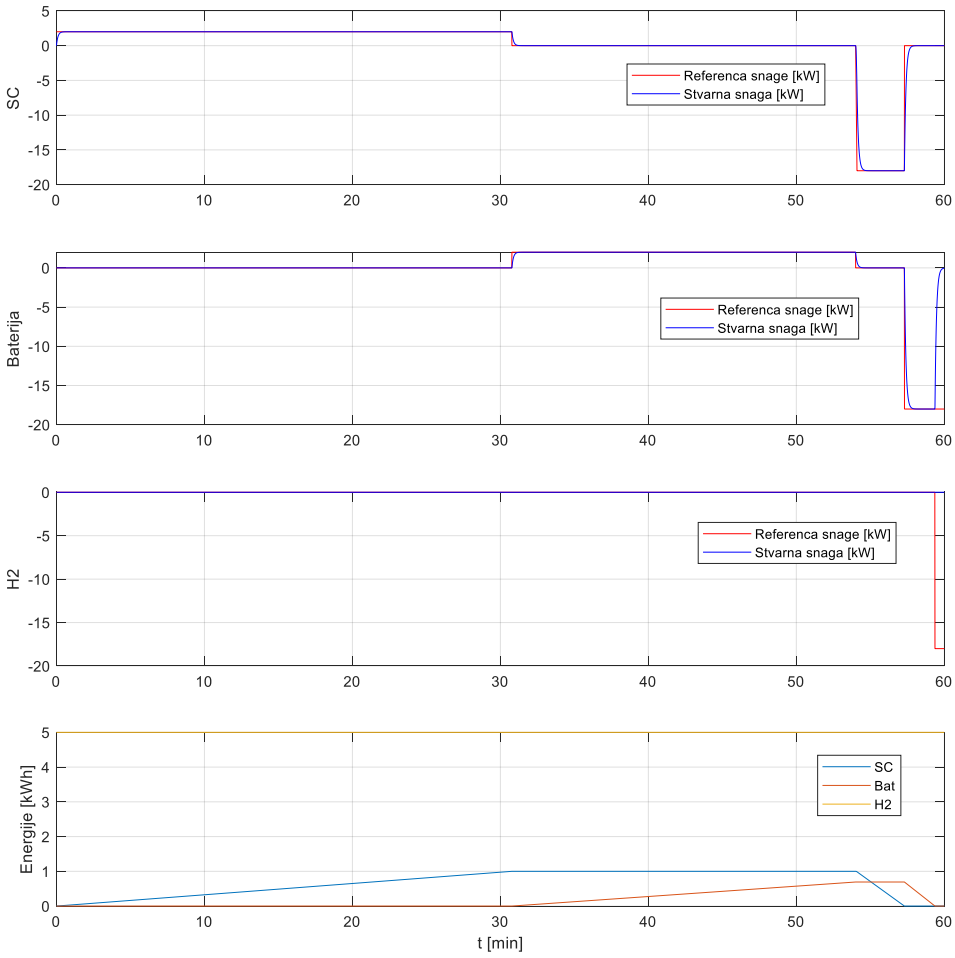


Slika 6.15. Ovisnost energija superkondenzatora („SC“), baterije i vodika („H2“) o snagama punjenja i pražnjenja u intervalu od jednog sata. SC ima snagu 1 kW, baterija 2 kW, a početna napunjenost vodika je 5 kWh. Snaga proizvodnje je postavljena na 10% maksimalne snage

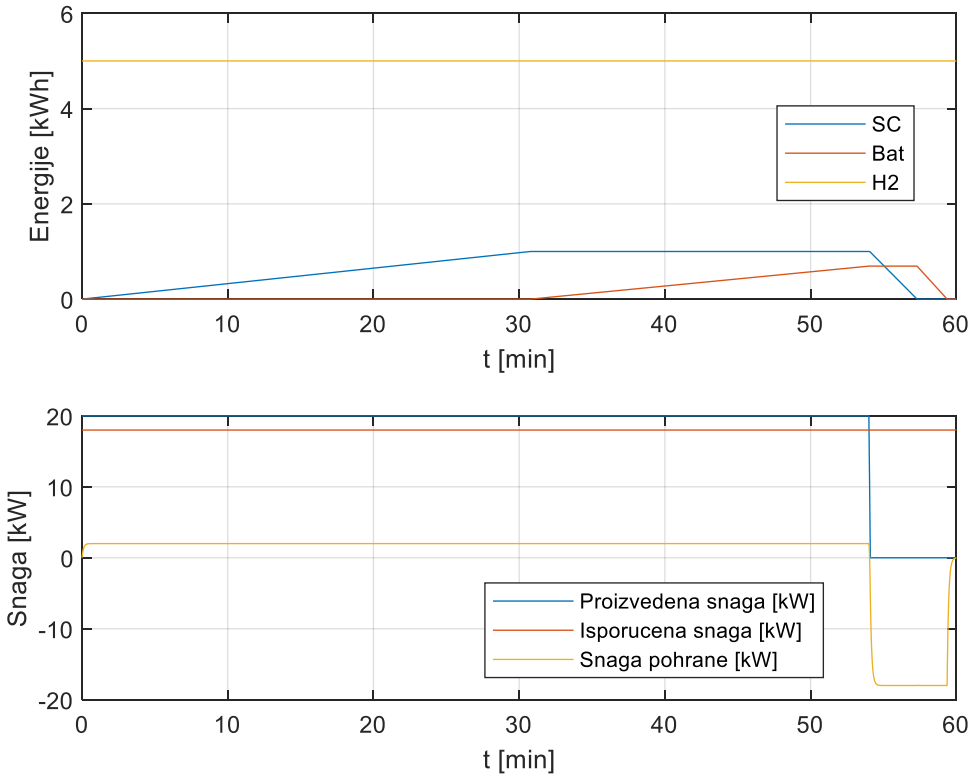


Slika 6.16. Ovisnost iznosa energija pohrane o snazi. SC ima snagu 1 kW, baterija 2 kW, a početna napunjenost vodika je 5 kWh. Snaga proizvodnje postavljena je na 10% maksimalne snage

U sljedećem slučaju su prikazani omjeri snaga i energija u slučaju kad je snaga proizvodnje postavljena na 90% maksimalne snage (54 minute je snaga proizvodnje postavljena na 20 kW, a 6 minuta snaga proizvodnje je 0 kW). U tom slučaju srednja snaga proizvodnje je 18 kW. Optimiranjem je određena granična vrijednost energije baterije 0,0000625 kWh i iznos učinkovitosti 0,2778.



Slika 6.17. Ovisnosti energija superkondenzatora („SC“), baterije i vodika („H2“) o snagama punjenja i pražnjenja u intervalu od jednog sata. SC ima snagu 1 kW, baterija 2 kW, a početna napunjenost vodika je 5 kWh. Snaga proizvodnje postavljenu je 90% maksimalne snage



Slika 6.18. Ovisnost energija pohrane o snazi (SC ima snagu 1 kW, baterija 2 kW, a početna napunjenost vodika je 5 kWh). Snaga proizvodnje postavljena je na 90% maksimalne snage

6.5. *Zaključne napomene*

Prema rezultatima iz prošlog poglavlja je vidljivo da je pronađen optimalan iznos granične razine energije za pohranu u bateriju kako bi se kompenzirala karakteristika vodika kao pohrane koja zahtijeva određeno vrijeme prije upotrebe, za što je u ovom modelu pretpostavljeno da traje jednu minutu. Predloženi model moguće je koristiti za različite oblike ulazne snage. Prema Prilogu F prikazano je da jedan ciklus optimiranja traje nešto više od 140 sekundi, što znači da je korekciju predikcije zasjenjenja sunca opisane u prethodnom dijelu rada zajedno s optimiranjem moguće izvršavati svakih 10 minuta, kako bi se što preciznije moglo upravljati tokovima energije. Time se izbjegava situacija u kojoj je u sustavu pohranjena nedovoljna količina energije za isporuku, uz prioretizaciju pohrane koja imaju veću učinkovitost: superkondenzatora čija učinkovitost je 0,95, baterije 0,8 te vodika 0,25.

Može se zaključiti da predloženi model temeljen na kratkoročnom predviđanju sunčeve dozračenosti dobro opisuje ponašanje hibridnog sustava pohrane energije u mikromreži.

7. Zaključak

Korištenje obnovljivih izvora energije temeljenih na Sunčevoj energiji potiče se kroz razne inicijative na razini Europske unije posljednjih godina. Strategija energetskog razvoja Hrvatske do 2030. s pogledom na 2050. godinu predviđa znatno veći udjel proizvodnje energije iz obnovljivih izvora. U razdoblju do 2030. godine planirano je povećanje udjela obnovljivih izvora u odnosu na potrošnju na barem 32% s potencijalnim povećanjem do 36,4%, dok bi do 2050. godine taj udjel trebao iznositi 65%. [47].

Obnovljivi izvori energije temeljeni na Sunčevoj energiji i fotonaponskim ćelijama proizvedenu energiju moraju skladištiti u pohrane s različitim karakteristikama tijekom perioda kad je Sunce vidljivo na nebu i nije zaklonjeno oblacima, kako bi se mogla koristiti u periodima kad je Sunce zaklonjeno oblacima ili nije vidljivo na obzoru. Predviđanje razine sunčeve dozračenosti omogućava učinkovito upravljanje skladištenjem proizvedene energije u pohrane koje se razlikuju po pitanju učinkovitosti, kapacitetu i brzini odziva. U sklopu modela korištenog u ovoj disertaciji korištena su tri pohrane: superkondenzator, baterija i vodik. Superkondenzator je zbog svoje brzine odziva i učinkovitošću od 0,95 odabran kao pohrana s najvećom prednosti, ali zbog visoke cijene ima ograničen kapacitet. Baterija s učinkovitošću od 0,8 se koristi nakon što se napuni ili isprazni superkondenzator, a vodik, iako ima nisku učinkovitost od 0,25 omogućava pohranjivanje gotovo neograničene količine energije.

Algoritmom implementiranim u programskom jeziku Java temeljenom na JavaCV biblioteci obrađuju se slike neba snimljene u razmaku od pet sekundi. Na njima se prvo određuju lokacije Sunca i oblaka, njihove konture i točke centroida, nakon čega se određuju smjerovi i brzine njihovih vektora kretanja.

Rezultati istraživanja u ovoj disertaciji pokazuju da se položaji oblaka mogu odrediti s obzirom na položaj Sunca ako su oblici kontura aproksimirani pravokutnikom s minimalnom površinom koja opisuje njihove konture. Aproksimacija oblika pravokutnika nema značajnog utjecaja na procjenu početka i kraja sjene oblaka nad suncem. Jednostavnost i brzina opisanog algoritma opisanog u ovom istraživanju čine pristup direktnog sunčevog zračenja prikladnim za upotrebu u mikromrežama s fotonaponskim sustavom. Pogreška izmjerena između predviđene dozračenosti i one u

stvarnom vremenu može se koristiti kao sigurnosno ograničenje pri projektiranju sustava za pohranu. Predviđena količina proizvedene energije može se smanjiti kako bi se izbjeglo pogrešno optimistično predviđanje koje može utjecati na stabilnost rada mikromreže. Pogrešku nije moguće utvrditi *on-line*, već na temelju iskustva stečenog mjerenjima i različitih razdoblja tijekom dana.

Prezentirani rezultati pokazuju da se kratkoročna predviđanja upotrebom kamere instalirane na tlu i upotreba procijenjene razine dozračenosti sunca u kombinaciji s prosječnom satnom prognozom sunčevog zračenja koju daje meteorološki servis mogu koristiti za predviđanje količine direktne sunčeve dozračenosti. Kratkoročno predviđanje izvodi se čim se oblak pojavi na slici neba koju je zabilježila kamera, tako da se nakon određenog vremenskog razdoblja (1 minuta ili 12 slika snimljenih svakih pet sekundi), postupak procjene može automatski započeti i traje nekoliko sekundi za sljedećih sat vremena. Jedino ograničenje je vidljivost sunca. Proces procjene može započeti kad se detektira početna pozicija sunca i oblaka na nebu.

Optimiranjem je uspješno određena granična razina energije koja mora biti pohranjena u bateriji kako bi se kompenzirao vremenski period do početka upotrebe vodika, iskorištavajući različite vrste pohrane energije poredanim po učinkovitosti. Najveću učinkovitost ima superkondenzator i u njega se energija najprije pohranjuje i iz njega prazni (0,95), zatim baterija koja se puni i prazni nakon superkondenzatora (ima učinkovitost 0,8) te na kraju vodik (s učinkovitošću od 0,25).

Budući rad uključivat će usporedbu s mjernim podacima prikupljenim piranometrom i uključivat će sva razdoblja u godini, uključujući jesen, zimu, proljeće i ljeto kako bi se pokrili svi mogući tipovi oblaka. Opcije, poput prilagodbe sustava različitim godišnjim dobima i vrstama oblaka, mogu se nadograditi komponentom strojnog učenja koja zamjenjuje ručni ljudski rad i poboljšava autonomiju sustava.

Literatura

[1] H. Dhrif *et al.*, "Learning and estimating whole sky visible, VNIR, SWIR radiance distributions from a commercial camera," *2019 IEEE PES GTD Gd. Int. Conf. Expo. Asia, GTD Asia 2019*, no. September, pp. 583–588, 2018, doi: 10.1117/12.2321295.

[2] T. Zhu, Y. Guo, C. Wang, and C. Ni, "Inter-hour forecast of solar radiation based on the structural equation model and ensemble model," *Energies*, vol. 13, no. 17, 2020, doi: 10.3390/en13174534.

[3] C. Pasion, T. Wagner, C. Koschnick, S. Schuldt, J. Williams, and K. Hallinan, "Machine learning modeling of horizontal photovoltaics using weather and location data," *Energies*, vol. 13, no. 10, pp. 1–14, 2020, doi: 10.3390/en13102570.

[4] J. Wang, Z. Qian, J. Wang, and Y. Pei, "Hour-ahead photovoltaic power forecasting using an analog plus neural network ensemble method," *Energies*, vol. 13, no. 12, pp. 1–17, 2020, doi: 10.3390/en13123259.

[5] J. Wojtkiewicz, M. Hosseini, R. Gottumukkala, and T. L. Chambers, "Hour-ahead solar irradiance forecasting using multivariate gated recurrent units," *Energies*, vol. 12, no. 21, pp. 1–13, 2019, doi: 10.3390/en12214055.

[6] M. Aslam, J. M. Lee, H. S. Kim, S. J. Lee, and S. Hong, "Deep learning models for long-term solar radiation forecasting considering microgrid installation: A comparative study," *Energies*, vol. 13, no. 1, 2019, doi: 10.3390/en13010147.

[7] G. Andrianakos, D. Tsourounis, S. Oikonomou, D. Kastaniotis, G. Economou, and A. Kazantzidis, "Sky Image forecasting with Generative Adversarial Networks for cloud coverage prediction," *10th Int. Conf. Information, Intell. Syst. Appl. IISA 2019*, no. July, pp. 1–7, 2019, doi: 10.1109/IISA.2019.8900774.

[8] G. Terrén-Serrano and M. Martínez-Ramón, "Processing of Global Solar Irradiance and Ground-Based Infrared Sky Images for Very Short-Term Solar Forecasting," no. January, 2021, [Online]. Available: http://arxiv.org/abs/2101.08694.

[9] C. N. Obiora, A. Ali, and A. N. Hasan, "Estimation of Hourly Global Solar Radiation Using Deep Learning Algorithms," *11th Int. Renew. Energy Congr. IREC 2020*, no. April, 2020, doi: 10.1109/IREC48820.2020.9310381.

[10] N. Maitanova *et al.*, “A machine learning approach to low-cost photovoltaic power prediction based on publicly available weather reports,” *Energies*, vol. 13, no. 3, 2020, doi: 10.3390/en13030735.

[11] Y. Kwon, A. Kwasinski, and A. Kwasinski, “Solar irradiance forecast using naïve bayes classifier based on publicly availableweather forecasting variables,” *Energies*, vol. 12, no. 8, 2019, doi: 10.3390/en12081529.

[12] V. Lešić, A. Martinčević, and M. Vašak, “Modular energy cost optimization for buildings with integrated microgrid,” *Appl. Energy*, vol. 197, pp. 14–28, 2017, doi: 10.1016/j.apenergy.2017.03.087.

[13] V. M. Miñambres-Marcos, M. Á. Guerrero-Martínez, F. Barrero-González, and M. I. Milanés-Montero, “A grid connected photovoltaic inverter with battery-supercapacitor hybrid energy storage,” *Sensors (Switzerland)*, vol. 17, no. 8, 2017, doi: 10.3390/s17081856.

[14] A. Yu, V. Chabot, and J. Zhang, “Electrochemical supercapacitors for energy storage and delivery: Fundamentals and applications,” *Electrochem. Supercapacitors Energy Storage Deliv. Fundam. Appl.*, pp. 1–355, 2017, doi: 10.1201/b14671.

[15] S. Park, Y. Kim, N. J. Ferrier, S. M. Collis, R. Sankaran, and P. H. Beckman, “Article prediction of solar irradiance and photovoltaic solar energy product based on cloud coverage estimation using machine learning methods,” *Atmosphere (Basel).*, vol. 12, no. 3, 2021, doi: 10.3390/atmos12030395.

[16] Y. Miyazaki, Y. Kameda, and J. Kondoh, “A Power-Forecasting Method for Geographically Distributed PV Power Systems using Their Previous Datasets,” *Energies*, vol. 12, no. 24, 2019, doi: 10.3390/en12244815.

[17] A. Kazantzidis, P. Tzoumanikas, P. Blanc, P. Massip, S. Wilbert, and L. Ramirez-Santigosa, “Short-term forecasting based on all-sky cameras,” *Renew. Energy Forecast. From Model. to Appl.*, no. December, pp. 153–178, 2017, doi: 10.1016/B978-0-08-100504-0.00005-6.

[18] G. Andrianakos *et al.*, “Inter-hour forecast of solar radiation based on the structural equation model and ensemble model,” *Energies*, vol. 13, no. 17, pp. 1–14, 2020, doi: 10.3390/en13174534.

[19] M. Schroedter-Homscheidt, M. Kosmale, and Y. M. Saint-Drenan, “Classifying

direct normal irradiance 1-minute temporal variability from spatial characteristics of geostationary satellite-based cloud observations," *Meteorol. Zeitschrift*, vol. 29, no. 2, pp. 131–145, 2020, doi: 10.1127/METZ/2020/0998.

[20] X. Li, Z. Lu, Q. Zhou, and Z. Xu, "A Cloud detection algorithm with reduction of sunlight interference in ground-based sky images," *Atmosphere (Basel).*, vol. 10, no. 11, 2019, doi: 10.3390/atmos10110640.

[21] J. Yang *et al.*, "A total sky cloud detection method using real clear sky background," *Atmos. Meas. Tech.*, vol. 9, no. 2, pp. 587–597, 2016, doi: 10.5194/amt-9-587-2016.

[22] A. Cazorla, C. Husillos, M. Antón, and L. Alados-Arboledas, "Multi-exposure adaptive threshold technique for cloud detection with sky imagers," *Sol. Energy*, vol. 114, no. March 2018, pp. 268–277, 2015, doi: 10.1016/j.solener.2015.02.006.

[23] A. Radovan and Z. Ban, "Prediction of HSV color model parameter values of cloud movement picture based on artificial neural networks," *2018 41st Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2018 - Proc.*, no. May 2018, pp. 1110–1114, 2018, doi: 10.23919/MIPRO.2018.8400202.

[24] S. Dev, Y. H. Lee, and S. Winkler, "Color-Based Segmentation of Sky/Cloud Images from Ground-Based Cameras," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 10, no. 1, pp. 231–242, 2017, doi: 10.1109/JSTARS.2016.2558474.

[25] P. Tuominen and M. Tuononen, "Cloud detection and movement estimation based on sky camera images using neural networks and the Lucas-Kanade method," *AIP Conf. Proc.*, vol. 1850, 2017, doi: 10.1063/1.4984528.

[26] J. O. Kamadinata, T. L. Ken, and T. Suwa, "Global solar radiation prediction methodology using artificial neural networks for photovoltaic power generation systems," *SMARTGREENS 2017 - Proc. 6th Int. Conf. Smart Cities Green ICT Syst.*, no. Smartgreens, pp. 15–22, 2017, doi: 10.5220/0006248700150022.

[27] P. Sukič and G. Štumberger, "Intra-minute cloud passing forecasting based on a low cost IoT sensor—a solution for smoothing the output power of PV power plants," *Sensors (Switzerland)*, vol. 17, no. 5, 2017, doi: 10.3390/s17051116.

[28] H. S. Jang, K. Y. Bae, H. S. Park, and D. K. Sung, "Solar Power Prediction

Based on Satellite Images and Support Vector Machine," *IEEE Trans. Sustain. Energy*, vol. 7, no. 3, pp. 1255–1263, 2016, doi: 10.1109/TSTE.2016.2535466.

[29] R. Chauvin, J. Nou, S. Thil, and S. Grieu, "Cloud motion estimation using a sky imager," *AIP Conf. Proc.*, vol. 1734, no. May 2016, 2016, doi: 10.1063/1.4949235.

[30] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for Easily Calibrating Omnidirectional Cameras," *2006 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2006, doi: 10.1109/IROS.2006.282372.

[31] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion," *Proc. Fourth IEEE Int. Conf. Comput. Vis. Syst.*, no. Icvs, 2006.

[32] M. Rufli, D. Scaramuzza, and R. Siegwart, "Automatic detection of checkerboards on blurred and distorted images," *2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS*, pp. 3121–3126, 2008, doi: 10.1109/IROS.2008.4650703.

[33] G. Bradski and A. Kaehler, *Learning OpenCV*. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., 2008.

[34] "PYPL PopularitY of Programming Language," 2021. https://pypl.github.io/PYPL.html.

[35] M. C. S. J. Gomes, L. Velho, *Computer Graphics: Theory and Practice*. CRC Press, 2012.

[36] A. Radovan and Ž. Ban, "Predictions of cloud movements and the sun cover duration," *2014 37th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2014 - Proc.*, no. May, pp. 1210–1215, 2014, doi: 10.1109/MIPRO.2014.6859752.

[37] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 4, pp. 532–550, 1987, doi: 10.1109/TPAMI.1987.4767941.

[38] "OpenCV Structural Analysis and Shape Descriptors." https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html (accessed May 01, 2021).

[39] M. Vasak and G. Kujundzic, "Comparison of battery management approaches for energy flow optimization in microgrids," *Proc. - 2016 IEEE Int. Power Electron.*

Motion Control Conf. PEMC 2016, pp. 348–353, 2016, doi: 10.1109/EPEPEMC.2016.7752022.

[40] M. Gulin, M. Vasak, and M. Baotic, "Analysis of microgrid power flow optimization with consideration of residual storages state," *2015 Eur. Control Conf. ECC 2015*, pp. 3126–3131, 2015, doi: 10.1109/ECC.2015.7331014.

[41] Y. Riffonneau, S. Bacha, F. Barruel, and S. Ploix, "Optimal power flow management for grid connected PV systems with batteries," *IEEE Trans. Sustain. Energy*, vol. 2, no. 3, pp. 309–320, 2011, doi: 10.1109/TSTE.2011.2114901.

[42] A. H. Fathima and K. Palanisamy, "Optimization in microgrids with hybrid energy systems - A review," *Renew. Sustain. Energy Rev.*, vol. 45, pp. 431–446, 2015, doi: 10.1016/j.rser.2015.01.059.

[43] X. Tan, Q. Li, and H. Wang, "Advances and trends of energy storage technology in Microgrid," *Int. J. Electr. Power Energy Syst.*, vol. 44, no. 1, pp. 179–191, 2013, doi: 10.1016/j.ijepes.2012.07.015.

[44] M. B. Burnett and L. J. Borle, "A power system combining batteries and supercapacitors in a solar/hydrogen hybrid electric vehicle," *2005 IEEE Veh. Power Propuls. Conf. VPPC*, vol. 2005, pp. 709–715, 2005, doi: 10.1109/VPPC.2005.1554636.

[45] M. Faisal, M. A. Hannan, P. J. Ker, A. Hussain, M. Bin Mansor, and F. Blaabjerg, "Review of energy storage system technologies in microgrid applications: Issues and challenges," *IEEE Access*, vol. 6, no. 1, pp. 35143–35164, 2018, doi: 10.1109/ACCESS.2018.2841407.

[46] S. A. Konstantinopoulos, A. G. Anastasiadis, G. A. Vokas, G. P. Kondylis, and A. Polyzakis, "Optimal management of hydrogen storage in stochastic smart microgrid operation," *Int. J. Hydrogen Energy*, vol. 43, no. 1, pp. 490–499, 2018, doi: 10.1016/j.ijhydene.2017.06.116.

[47] "Fond za zaštitu okoliša i energetsku učinkovitost." https://www.fzoeu.hr/hr/obnovljivi-izvori-energije/7573 (accessed Jun. 17, 2022).

Popis slika

Popis tablica

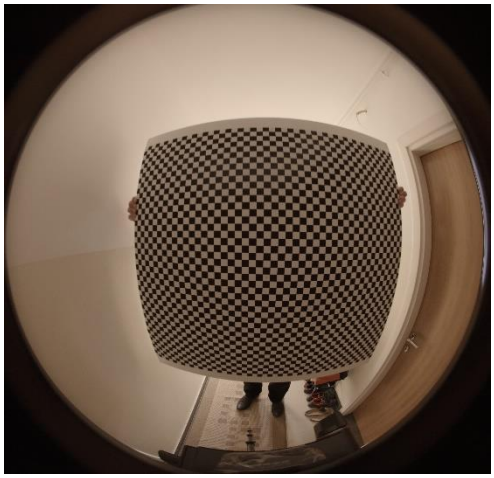
Prilozi

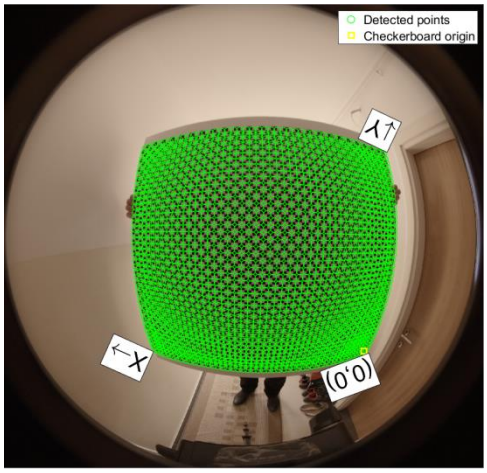
Prilog A. Matlab skripta za ispravljanje efekta distorzije na slikama snimljenim širokokutnom kamerom

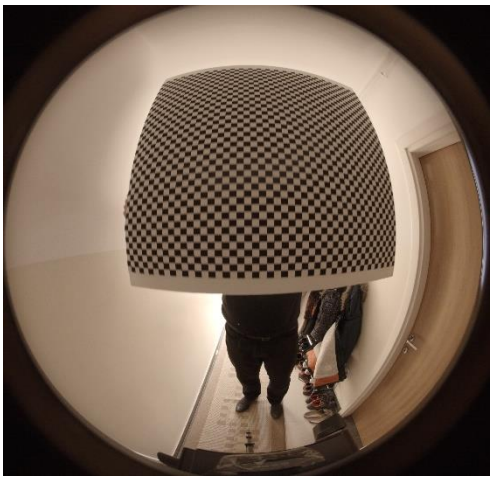
```
% Definiranje lokacije slika šahovnice u različitim položajima
imageFileNames = {
'D:\Doktorat\Slike šahovnice\GF124909.JPG',...
'D:\Doktorat\Slike šahovnice\GF124913.JPG',...
'D:\Doktorat\Slike šahovnice\GF124921.JPG',...
'D:\Doktorat\Slike šahovnice\GF124931.JPG',...
'D:\Doktorat\Slike šahovnice\GF124932.JPG',...
'D:\Doktorat\Slike šahovnice\GF124957.JPG',...
'D:\Doktorat\Slike šahovnice\GF124986.JPG',...
'D:\Doktorat\Slike šahovnice\GF135063.JPG',...
'D:\Doktorat\Slike šahovnice\GF135068.JPG',...
'D:\Doktorat\Slike šahovnice\GF135084.JPG',...
'D:\Doktorat\Slike šahovnice\GF135102.JPG',...
   };
% Detektiranje šahovske ploče na slikama
[imagePoints, boardSize, imagesUsed] =
detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Učitavanje prve slike u svrhu određivanja veličine slike
originalImage = imread(imageFileNames{1});
[mrows, ncols, ~] = size(originalImage);

squareSize = 20;  % definiranje veličine kvadratića šahovske
                  % ploče (u mm)
% Generiranje prostornih koordinata vrhova kvadratića šahovske
% ploče
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Kalibriranje kamere korištenje „fisheye“ parametara
[cameraParams, imagesUsed, estimationErrors] =
estimateFisheyeParameters(imagePoints, worldPoints, ...
    [mrows, ncols], ...
'EstimateAlignment', false, ...
'WorldUnits', 'millimeters');

% Prikazivanje pogrešaka projekcije
h1=figure; showReprojectionErrors(cameraParams);

% Vizualizacija uzoraka lokacija i pozicije šahovskih ploča
h2=figure; showExtrinsics(cameraParams, 'CameraCentric');

% Prikazivanje pogrešaka estimiranih parametara
displayErrors(estimationErrors, cameraParams);
```

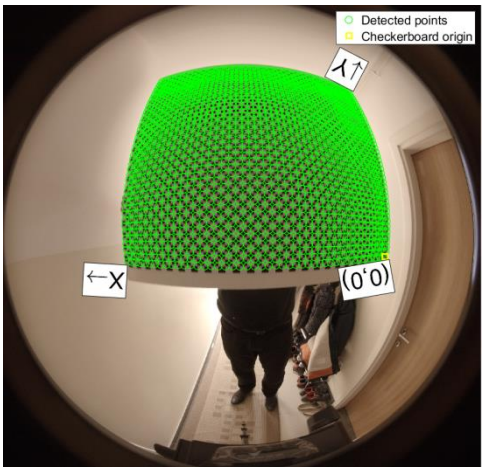
```
% Postavljanje mape s iskrivljenim slikama
myFolder = 'D:\Doktorat\IskrivljeneSlike';

% Postavljanje mape za spremanje neiskrivljenih slika
myNewFolder = 'D:\Doktorat\NeiskrivljeneSlike\';

% Učitavanje JPG slika nema
myFiles = dir(fullfile(myFolder, '*.JPG'));

% Petlja koja prolazi po svim slikama
for k = 1:length(myFiles)

  % Dohvaćanje naziva slike
baseFileName = myFiles(k).name;

% Dohvaćanje naziva slike zajedno s lokacijom
fullFileName = fullfile(myFolder, baseFileName);

  % Ispis lokacije slike koja se trenutno obrađuje
fprintf(1, 'Nowreading %s\n', fullFileName);

  % Učitavanje slike
originalImage = imread(fullFileName);

% Obrađivanje slike i odstranjivanje efekata distorzije
  % (parametar „0.39“ daje najbolje rezultate)
undistortedImage = undistortFisheyeImage(originalImage,
cameraParams.Intrinsics, 'ScaleFactor', .39);

% Postavljanje imena slike bez efekta distorzije
newFullFileName =fullfile(myNewFolder, baseFileName);

% Spremanje slike bez efekta distorzije
imwrite(undistortedImage,newFullFileName);
end
```

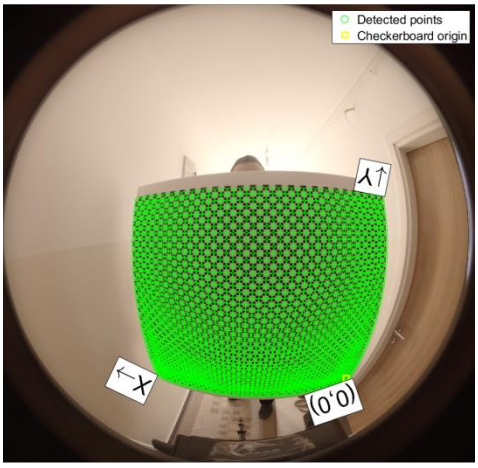
Prilog B. Slike šahovske ploče snimljene pomoću širokokutne kamere GoProFusion i pripadajuće slike s detektiranim kvadratićima na šahovskoj ploči

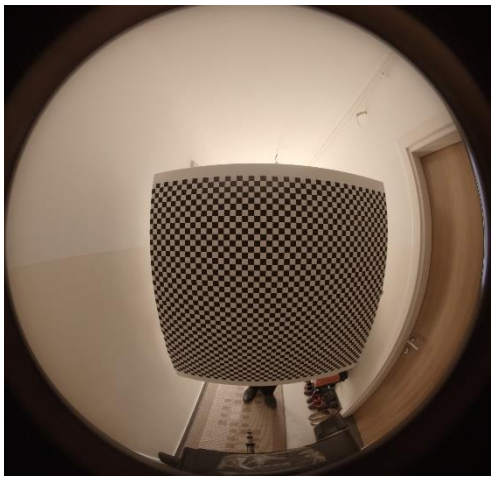


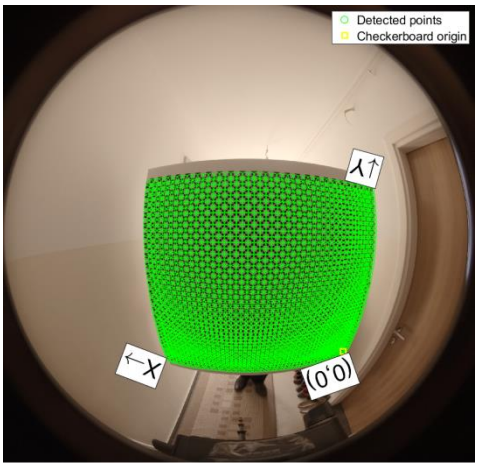


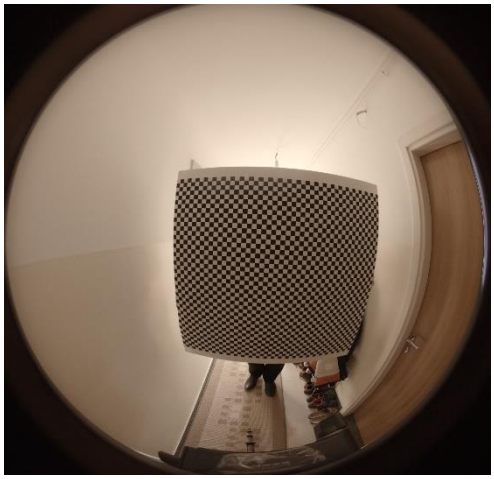


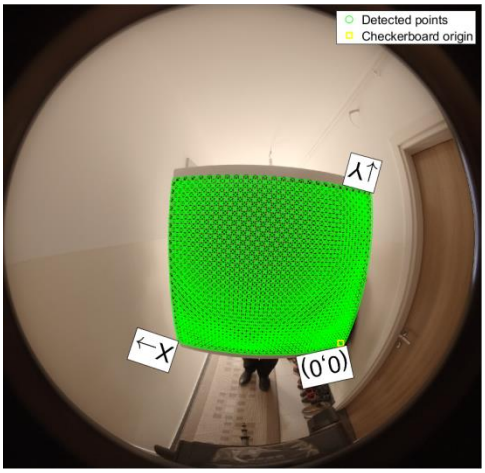


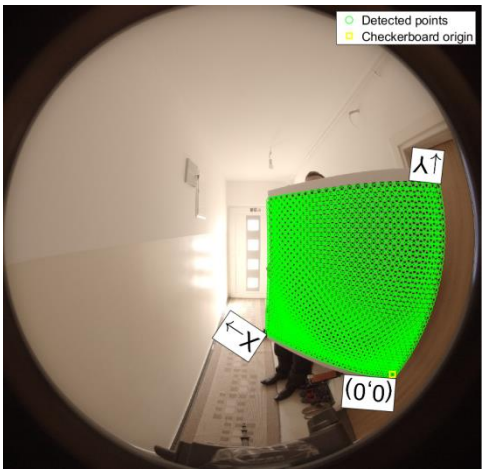


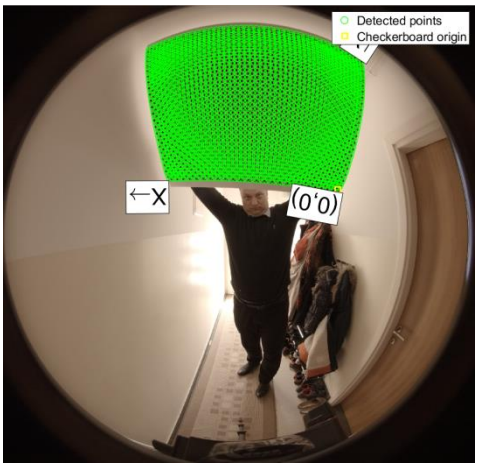


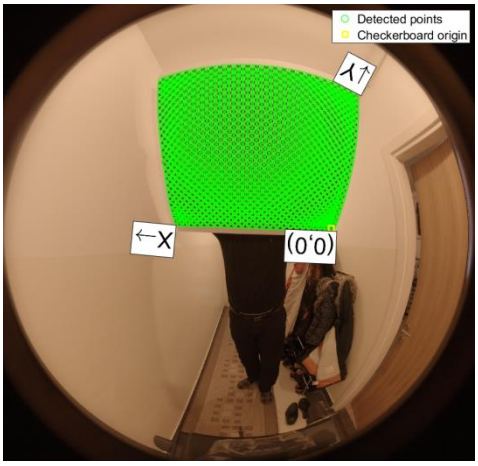

Detected points
Checkerboard origin
Y
X
(0,0)

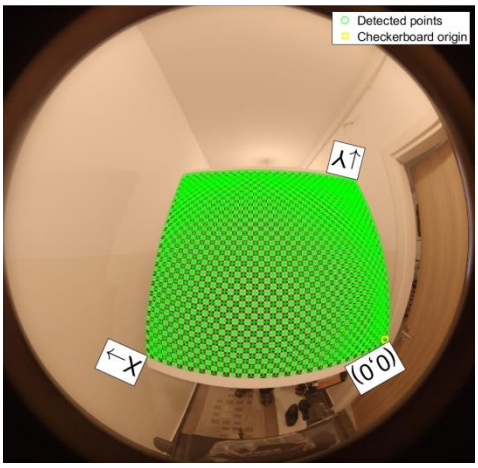


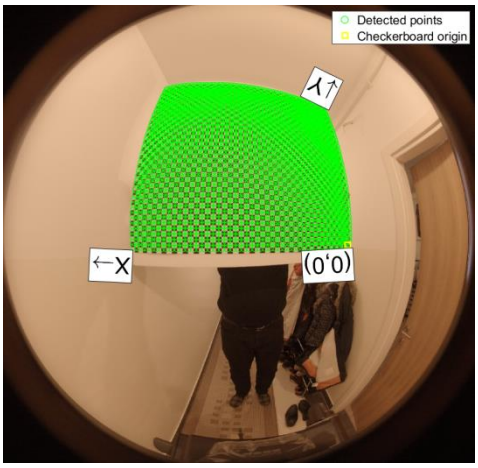

Detected points
Checkerboard origin
Y
X
(0,0)

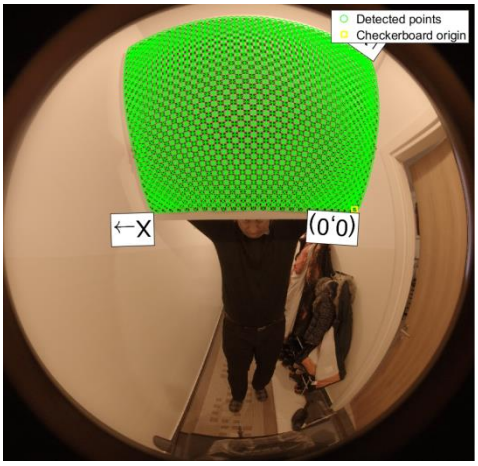

Detected points
Checkerboard origin
Y
X
(0,0)


Detected points
Checkerboard origin
X
(0,0)


Detected points
Checkerboard origin
Y
X
(0,0)


Detected points
Checkerboard origin
Y
X
(0,0)


Detected points
Checkerboard origin
Y
X
(0,0)


Detected points
Checkerboard origin
X
(0,0)

Prilog C. Programski kod u jeziku Java koji koristi JavaCV biblioteku te generira krivulju predviđene dozračenosti

```
package hr.fer.phd.radovan;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.TimeZone;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfPoint2f;
import org.opencv.core.Point;
import org.opencv.core.RotatedRect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.imgproc.Moments;

/**
 * Generates data for irradiance estimation curve based on sun and cloud
 * tracking data. Wide-angle ground-based camera GoPro Fusion was used to
  *obtain images of the sky.
 *
 * @author Aleksander Radovan
 *
 */
public class IrradianceEstimationCurveGenerator {

	// data structures for saving contours and centroid points of clouds
	// and the sun
	private static List<List<Point>>cloudCentroids = new ArrayList<>();
	private static List<List<MatOfPoint>>
		cloudContours = new ArrayList<>();
	private static List<Point>sunCentroids = new ArrayList<>();

	/**
	 * Starts the algorithm for generating the data for irradiance
	 * estimation curve.
	 *
	 * @param args-command line arguments (not used)
	 * @throws FileNotFoundException thrown if sky images not found
	 */
	Public static void main(String[] args) throws FileNotFoundException {
```

```
// fetching the start time for measuring time duration for the
// algorithm
long start = System.currentTimeMillis();

// loading OpenCV DLL files for Windows operating system,
// VM parameterneeds to be set:
// (-Djava.library.path=C:\DEV\LIBs\OpenCV-
/ 4.3.0\opencv\build\java\x64)
System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

// data structure for saving coverage records
List<List<CoverageRecord>> data = new ArrayList<>();

// loop that processes all ordinal numbers of folders with
// ordinal numbers sky images created by the GoPro Fusion wide-
// angle ground-based camera
for (int i = 205; i <= 205; i++) {

	// creating original image path to get information
	// about the original time of taking the image
	String originalmagePath =
		"D:\\Fusion\\15-07-2020\\" + i + "GFRNT\\";

	// creating the image path of undirstorted image
	String imagePath =
		"D:\\Fusion\\15-07-2020\\" + i +
			"GFRNT\\undistorted";

	// fetching all the files from the folder of
	// undistorted images
	File[] listOfFiles = new File(imagePath).listFiles();

	// declaring the variable that will contain the last known
	// sun centroid point in case of cloud covering the sun
	Point lastKnownSunCentroid = null;

	// image ordinal number
	long imageNumber = 0;

	// loop that fetches all the image files inside of the
	// current folder
	for (File imageFile : listOfFiles) {

		// if the file system item is not an image
		// (in case of nested folder), skip it
		if (imageFile.isFile() == false) {
			continue;
		}

		// fetch the image number from the image name
		// created by the camera (for example,
		// "GF018382.JPG"),the number between the third
		// (index 2) and the ninth (index 8) is being
		// extracted and parsed
		IntegerimageOrdinalNumber =
			Integer.parseInt(
				imageFile.getName().substring(2, 8));
```

```
// original file that represents the sky image with
// the full path
File originalImageFile = new File(originalmagePath
      + "GF0" + imageOrdinalNumber + ".JPG");

// printing out the current image name being
// processed
System.out.println("Processing " + imagePath
      + "\\" + imageFile.getName());

// getting the time when the original image was
// taken from the camera
LocalDateTimetriggerTime =
      LocalDateTime.ofInstant(
            Instant.ofEpochMilli(
            originalImageFile.lastModified()),
            TimeZone.getDefault().toZoneId());

// printing out the current time
System.out.println(triggerTime);

// creating the object for the centroid of thesun
PointsunCentroid = new Point();

// creating the OpenCV image container for
// detecting the sun on the sky
Mat originalSkyImageMat = Imgcodecs.imread(
      imageFile.getAbsolutePath());

// creating the OpenCV  imagecontainer for
// detecting clouds on the sky
Mat originalCloudsImageMat = Imgcodecs.imread(
      imageFile.getAbsolutePath());

// creating the OpenCV image  container for HSV
// color model of the image with the
// sun on the sky
Mat hsvSkyImage = new Mat();

// creating the OpenCV image container for HSV
// color model of the image with
// clouds on the sky
Mat hsvCloudsImage = new Mat();

// creating the OpenCV image container for creating
// the mask for the shape of
// the sun
Mat maskForSun = new Mat();

// creating the  OpenCV image container for
// creating the mask for the shape of clouds
Mat maskForClouds = new Mat();

// converting the image with the sky from RGB to
// HSV color model
Imgproc.cvtColor(originalSkyImageMat, hsvSkyImage,
```

```
        Imgproc.COLOR_BGR2HSV);

// converting the image with the sky from RGB to
//HSV color model
Imgproc.cvtColor(originalCloudsImageMat,
        hsvCloudsImage,
                Imgproc.COLOR_BGR2HSV);

// creating the OpenCV image container for the
//results of mophological operation
// erosion on the image with the sky
Mat morphOutputForSky = new Mat();

// creating the OpenCV image container for the
// results of mophological operation
// erosion and dilation on the image with clouds
Mat morphOutputForClouds = new Mat();

// setting the minimum and maximum values for
// conversion applying threshold
// filters on images with the sun and the clouds
ScalarminValuesForSun = new Scalar(0, 0, 240);
ScalarmaxValuesForSun = new Scalar(0, 0, 255);
ScalarminValuesForClouds = new Scalar(0, 0, 0);
ScalarmaxValuesForClouds =
        New Scalar(70, 70, 255);

// filtering the HSV images with defined scalar
// thresholds
Core.inRange(hsvSkyImage, minValuesForSun,
        maxValuesForSun, maskForSun);
Core.inRange(hsvCloudsImage, minValuesForClouds,
        maxValuesForClouds, maskForClouds);

// applying morphological operations of erosion and
// dilation to images with the
// sun and clouds
// structural element size for eroding the image of
// the sky is 30 x 30
// structural element size for eroding the image
// with clouds is 20 x 20
// structural element size for dilating the image
// with clouds is 30 x 30
Mat dilateElementForClouds =
        Imgproc.getStructuringElement(
                Imgproc.MORPH_RECT,
                        New Size(30, 30));
Mat erodeElementForSky =
        Imgproc.getStructuringElement(
                Imgproc.MORPH_RECT, new Size(50, 50));
Mat erodeElementForClouds =
        Imgproc.getStructuringElement(
                Imgproc.MORPH_RECT, new Size(20, 20));

// applying the erosion operation to the image
// with the sun and the clouds
Imgproc.erode(maskForSun, morphOutputForSky,
```

```
        erodeElementForSky);
Imgproc.erode(maskForClouds, morphOutputForClouds,
        erodeElementForClouds);

// applying the dilation operation to the image
// with the clouds
Imgproc.dilate(maskForClouds, morphOutputForClouds,
        dilateElementForClouds);

// creating a data structure for saving contours of
// the sun
List<MatOfPoint>contoursForSun =
        new ArrayList<>();

// creating the OpenCV image container for
// hierarchy levels of contours of the
// sun
Mat hierarchyForSky = new Mat();

// finding countours of the sun
Imgproc.findContours(morphOutputForSky,
        contoursForSun, hierarchyForSky,
        Imgproc.RETR_TREE,
        Imgproc.CHAIN_APPROX_SIMPLE);

// creating a data structure for saving contours of
// clouds
List<MatOfPoint>contoursForClouds = new
        ArrayList<>();

// creating the OpenCV image container for
// hierarchy levels of contours of
// clouds
Mat hierarchyForClouds = new Mat();

// finding countours of clouds
Imgproc.findContours(morphOutputForClouds,
        contoursForClouds, hierarchyForClouds,
        Imgproc.RETR_TREE,
        Imgproc.CHAIN_APPROX_SIMPLE);

// declaring the variables for sun contour area,
// sun rectangle area, sun contour
// index in the list
// of detected contours (if more shiny  contours on
// the sky detected), detected
// rectangle around
// the sun and the flag if the sun has been
// detected or if it's covered with
// clouds
double sunArea = 0;
double sunRectangleArea = 0;
int sunContourIndex = 0;
RotatedRectrotatedRectAroundSun = null;
Boolean sunDetected = false;

// declaring four points for rectangle around the
```

```
// sun contour
Point[] sunPoints = new Point[4];

// check if the sun has been detected on the sky
// image
if (contoursForSun.size() > 0) {

	// reseting the sun area to zero
	sunArea = 0;

	// loop that fetches all candidates for the
	// sun and saves the
	// area and the contour index with the
	// largest area (if more than
	// one shiny contours found on the image)
	for (int k = 0; k <contoursForSun.size();
		k++)
	{
		MatOfPointsunContour =
			contoursForSun.get(k);
		double newArea =
			Imgproc.contourArea(sunContour);
		if (newArea>sunArea) {
			sunArea = newArea;
			sunContourIndex = k;
		}
	}

	// drawing the contours of the  sun on the
	// image
	Imgproc.drawContours(originalSkyImageMat,
		contoursForSun, sunContourIndex,
		new Scalar(0, 0, 255),
		10, 2, hierarchyForSky);

	// calculating moments in order to determine
	// the sun contour
	Momentsmoments =
		Imgproc.moments(
		contoursForSun.get(sunContourIndex));
	sunCentroid.x = moments.get_m10() /
				moments.get_m00();
	sunCentroid.y = moments.get_m01() /
				moments.get_m00();

	// saving the centroid point of the detected
	// sun
	sunCentroids.add(sunCentroid);

	// if sun area has the minimum area of 5000
	// pixels, it  isconsidered
	// as successfuly detected sun and the last
	// known sun centroid is saved
	if (sunArea > 5000) {
		lastKnownSunCentroid = sunCentroid;
	}
```

```
                // setting the flag for successfuly detected
                // sun if any of the contours are
                // found
                if (sunArea > 0) {
                        sunDetected = true;
                }

                // drawing a circle that represents the
                // centroid of the sun
                Imgproc.circle(originalSkyImageMat,
                        sunCentroid, 10, new Scalar(0, 0,
                        255), 10);

                // determining the rotated rectangle shape
                // around the contour of the  sun
                rotatedRectAroundSun = Imgproc
                                .minAreaRect(
                                        new MatOfPoint2f(
                                        contoursForSun.get(
                                        sunContourIndex).toArray(
                                ));

                // calculating the area of the rotated
                // rectangle around the sun
                sunRectangleArea =
                        rotatedRectAroundSun.boundingRect()
                                .area();

                // initialing the array of points for
                // rectangle shape
                sunPoints = new Point[4];

                // setting the rectangle points around the
                // sun contour
                rotatedRectAroundSun.points(sunPoints);

                // loop that draws the rectangle with four
                // lines that connect four points
                // detected in previous steps
                for (int p = 0; p < 4; p++) {
                        Imgproc.line(originalSkyImageMat,
                                sunPoints[p],
                                sunPoints[(p + 1) % 4],
                                new Scalar(0, 0, 255),
                                5);
                }

        } else {
                // if contours of the sun are not detected,
                // the flag is set to false
                sunDetected = false;
        }

        // creating data structures for saving clouds
        // centroids and contours
        List<Point> listOfCentroidsForImage =
                new ArrayList<>();
```

```
List<MatOfPoint> listOfContours =
      new ArrayList<>();

// checking if any clouds are detected
if (contoursForClouds.size() > 0) {

      // loop for iterating through the cloud
      // contours
      for (int k = 0; k < contoursForClouds.size();
            k++)
      {

            // calculating area of the cloud
            // contour
            double cloudArea =
                  Imgproc.contourArea(
                  contoursForClouds.get(k));

            // if the area has a significant value
            // (at least 60% of the sun area)
            if (cloudArea> 3000) {

            // calculate the moments to
            // determine centroid points of the
            // cloud
                  MomentscloudMoments =
                  Imgproc.moments(
                  contoursForClouds.get(k));
                  PointcloudCentroid =
                        new Point();
                  cloudCentroid.x =
                        cloudMoments.get_m10() /
                        cloudMoments.get_m00();
                  cloudCentroid.y =
                        cloudMoments.get_m01() /
                        cloudMoments.get_m00();

                  // if the cloud position is
                  // outside the range  ofcovering
                  // the sun – ignore it
                  // (centroid of the cloud is on
                  // the possible path of the sun)
                  if (cloudCentroid.y> 600 &&
                        cloudCentroid.y< 2500)
                  {

                        // if the cloud centroid
                        // has "passed" the sun
                        // position – ignore  it
                        if ((sunCentroid.x + 300)
                              >cloudCentroid.x
                                    &&
                              (sunCentroid.y +
                              300)
                              >cloudCentroid.y)
                        {
```

```
// if the cloud centroid
// is too near the sun
// centroid - it's the
// sun's contour,
// so skip it
if (calculateDistance(
      sunCentroid,
cloudCentroid) > 100) {
// draw the contour of
// the cloud
Imgproc.drawContours(
originalCloudsImageMat,
contoursForClouds, k,
new Scalar(255, 0, 0),
10, 2,
hierarchyForClouds);

// calculate the bounding
// rectangle around the
// cloud
RotatedRect
rotatedRectAroundCloud =
Imgproc
.minAreaRect(
      new MatOfPoint2f(
      contoursForClouds
.get(k).toArray()));

// initialize the points
// that define the
// bounding rectangle
// around the cloud
Point[] points =
      new Point[4];

// set the rectangle
// points

rotatedRectAroundCloud
      .points(points);

// draw ther ectangle
// with four lines
for (int p = 0; p < 4;
      p++)
{
Imgproc.line(
originalCloudsImageMat,
points[p],
points[(p + 1) % 4],
new Scalar(0, 255, 0),
5);
}

// save cloud centroids
// and contours
listOfCentroidsForImage
```

```
                                        .add(cloudCentroid);

                                        listOfContours.add(
                                        contoursForClouds.get(
                                              k));
                                  }
                            }
                      }
                }
          }
}

// saving cloud contours and centroid points
cloudCentroids.add(listOfCentroidsForImage);
cloudContours.add(listOfContours);

// if twelve images are processed, the simulation of the next
// period can start
if (cloudContours.size() > 12) {

      // fetching the contours that were detected on the last
      // image and a minute ago (12 x 5 seconds frequency of
      // taking image = 60 seconds = 1 minute)
      List<MatOfPoint>firstPhaseContours =
            cloudContours.get(cloudContours.size() - 13);
      List<MatOfPoint>secondPhaseContours =
            cloudContours.get(cloudContours.size() - 1);

      // fetching the centroids that were detected on the
      // last image a minute ago
      Point firstSunCentroidPoint =
            sunCentroids.get(sunCentroids.size() - 13);
      Point secondSunCentroidPoint =
            sunCentroids.get(sunCentroids.size() - 1);

      // calculating the sun shift within one period of 5
      // seconds
      double sunShiftX = (secondSunCentroidPoint.x–
            firstSunCentroidPoint.x) / 12;
      double sunShiftY = (secondSunCentroidPoint.y–
            firstSunCentroidPoint.y) / 12;

      // drawing the vector of sun's movement
      Imgproc.circle(originalSkyImageMat,
            secondSunCentroidPoint, 1, new Scalar(0, 0, 255),
            10);

      Imgproc.arrowedLine(originalSkyImageMat,
            firstSunCentroidPoint, secondSunCentroidPoint,
            newScalar(0, 0, 255), 10, 3, 0);

      // initializing the second centroid index and point
      intsecondContourIndex = 0;
      Point secondCentroid = new Point();

      // loops that finds the nearest centroid point of the
      // cloud to determine the speed vector
```

```
for (MatOfPointsecondContour : secondPhaseContours) {

	// calculating the centroid of the cloud
	MomentssecondCentroidMoments =
		Imgproc.moments(secondContour);
	secondCentroid.x = secondCentroidMoments.get_m10()
		/ secondCentroidMoments.get_m00();
	secondCentroid.y = secondCentroidMoments.get_m01()
		/ secondCentroidMoments.get_m00();

	// initialization of thenearest centroid point
	PointnearestCentroid = new Point();

	// initializing the distance to the nearest contour
	// variable
	double distanceToNearestContour = -1;

	// loop that finds the nearest contour
	for (MatOfPointfirstContour : firstPhaseContours)
	{

		// initializing and determining the centroid
		// point of the contour
		PointfirstCentroid = new Point();
		MomentsfirstCentroidMoments =
			Imgproc.moments(firstContour);
		firstCentroid.x =
			firstCentroidMoments.get_m10() /
			firstCentroidMoments.get_m00();
		firstCentroid.y =
			firstCentroidMoments.get_m01() /
			firstCentroidMoments.get_m00();

		// calculating new distance between the
		// first and the second cloud centroid
		double newDistance =
			calculateDistance(secondCentroid,
			firstCentroid);

		// if distance ispositive, check if it's the
		// new nearest distance and saveit
		if (newDistance> 0) {
			if (distanceToNearestContour == -1) {
				distanceToNearestContour =
					newDistance;
				nearestCentroid = firstCentroid;
			} elseif (distanceToNearestContour>
				newDistance) {
				distanceToNearestContour =
					newDistance;
				nearestCentroid = firstCentroid;
			}
	}
}

// calculatethe cloud shift within five seconds
double shiftX = (secondCentroid.x - nearestCentroid.x) /
```

```
                                                        12;
double shiftY = (secondCentroid.y - nearestCentroid.y) /
                                                        12;

// initialize the contour of the cloud
MatOfPoint cloudContour =
      secondPhaseContours.get(secondContourIndex);

// if the contour is not the sun's contour (the centroid
// needs to be at least 100 pixels away)
// and it's still in the area of the sun'spath
if (calculateDistance(sunCentroid,secondCentroid) > 100
      && (secondCentroid.y> 300) && (secondCentroid.y<
                                                        2600))
{

// creating the OpenCV image container for drawing the
// cloud path
Mat cloudPathImage =
      Imgcodecs.imread(imageFile.getAbsolutePath());

// determining rotated rectangle around the cloud
RotatedRect rotatedRectAroundCloud
      = Imgproc.minAreaRect(new
            MatOfPoint2f(cloudContour.toArray()));

// initializing and drawing the rectangle points of the
// cloud
Point[] points = new Point[4];

rotatedRectAroundCloud.points(points);

for (int p = 0; p < 4; p++) {
      Imgproc.line(cloudPathImage, points[p],
      points[(p + 1) % 4], new Scalar(0, 255, 0), 5);
}

// drawing the rectangle points of the sun
for (int p = 0; p < 4; p++) {

      Imgproc.line(cloudPathImage, sunPoints[p],
            sunPoints[(p + 1) % 4],
            new Scalar(0, 0, 255), 5);
}

// initialization of variables for tracking the upper left
// corner of the bounding rectangle
doublemin_x = 0;
doublemin_y = 0;

// fetching the original time of taking the sky image
LocalTimecloudTime = LocalTime.ofInstant(
Instant.ofEpochMilli(originalImageFile.lastModified()),
TimeZone.getDefault().toZoneId());

// initializing the data structure for sun coverage
// records
```

```
List<CoverageRecord> coverageRecordList =
                    New ArrayList<>();

// declaring a variable that counts covered periods
Int periods = 0;

// loop that simulates the movement of clouds and the sun
// in the future checking the upper left corner of the
// bounding rectangle
// if it's still not fully visible on the sky image, the
// coordinates are outside of the image (x >= -200, y >= -
// 200) if it's left the sky image area, x > 3104, y >
// 3000 (image resolution)
while ((min_x>= -200 &&min_x<= 3104)
      && (min_y>= -200 &&min_y<= 3000)) {

      // creating the OpenCV image container for
      // intersection region of the cloud and the sun
      Mat intersectingRegion = new Mat();

      // fetching the status of the intersection (can be
      // "INTERSECT_FULL", "INTERSECT_PARTIAL" or
      // "INTERSECT_NONE")
      int status = Imgproc.rotatedRectangleIntersection(
            rotatedRectAroundSun,
            rotatedRectAroundCloud,intersectingRegion);

      // declaring the variable that holds
      // intersectionArea
      double intersectingArea = 0;

      // if the clouds rectangle intersects the sun's
      // rectangle
      if (status == Imgproc.INTERSECT_FULL
      || status == Imgproc.INTERSECT_PARTIAL) {

            // calculate the intersection area
            intersectingArea =
                  Imgproc.contourArea(intersectingRegion
                  );
      }

      // add the coverage record to the list

      coverageRecordList.add(
            newCoverageRecord(secondContourIndex,
            cloudTime,
            intersectingArea / sunRectangleArea));

      // increase the cloud time that is a foundation for
      // prediction current cloud time by five seconds
      // and increase corresponding minutes and hours i
      // if needed
      if (cloudTime.getMinute() + 1 > 59) {
            if (cloudTime.getHour() + 1 < 24) {
                  cloudTime =
                        LocalTime.of(cloudTime.getHour()
```

```
                        + 1, 0, cloudTime.getSecond());
        } else {
                cloudTime = LocalTime.of(0, 0,
                        cloudTime.getSecond());
        }
} elseif (cloudTime.getSecond() + 5 > 59) {
        cloudTime = LocalTime.of(cloudTime.getHour(),
                cloudTime.getMinute() + 1,
                cloudTime.getSecond() - 55);
} else {
        cloudTime = LocalTime.of(cloudTime.getHour(),
                cloudTime.getMinute(),
                cloudTime.getSecond() + 5);
}

// shifting the cloud's bounding rectangle by
// shiftX and shiftY amounts (in pixels)
for (int p = 0; p < 4; p++) {
        points[p].x = points[p].x + shiftX;
        points[p].y = points[p].y + shiftY;
}

// shifting the sun's bounding rectangle by shiftX
// and shiftY amounts (in pixels)
for (int p = 0; p < 4; p++) {
        sunPoints[p].x = sunPoints[p].x + sunShiftX;
        sunPoints[p].y = sunPoints[p].y + sunShiftY;
}

// determining the shifted rectangle position of
// the cloud
MatOfPoint2f cloudRectanglePoints =
        new MatOfPoint2f(points[0], points[1],
        points[2],points[3]);

// determining the area of the cloud's bounded
// rectangle
rotatedRectAroundCloud =
        Imgproc.minAreaRect(cloudRectanglePoints);

// determining the shifted rectangle position of
// the sun
MatOfPoint2f sunRectanglePoints =
        new MatOfPoint2f(sunPoints[0], sunPoints[1],
        sunPoints[2], sunPoints[3]);

// determining the area of the sun's bounded
// rectangle
rotatedRectAroundSun =
        Imgproc.minAreaRect(sunRectanglePoints);

// if the period variable hits a five minutes
// threshold (60 x 5 seconds = 300 seconds =  5
// minutes)
if (periods % 60 == 0) {

        // draw a rectangle of simulated position of
```

```
                        // the cloud in the future
                        for (int p = 0; p < 4; p++) {
                              Imgproc.line(cloudPathImage,
                              points[p], points[(p + 1) % 4],
                              newScalar(0, 0, 0), 5);
                        }

                        // put time text to the drawn rectangle on
                        // the sky image
                        Imgproc.putText(cloudPathImage,
                              cloudTime.format(
                                    DateTimeFormatter.ofPattern(
                                          "HH:mm:ss")),
                              newPoint(points[2].x, points[2].y -
                              50), 2, 2.0,newScalar(0, 0, 0), 2);
                  }

                  // fetch the upper left corner of the bounding
                  // rectangle to determine if the cloud
                  // needs to be considered as candidate for covering
                  // the sun
                  min_x = Arrays.stream(points).mapToDouble(
                        p ->p.x).sorted().min().getAsDouble();
                  min_y = Arrays.stream(points).mapToDouble(
                        p ->p.y).sorted().min().getAsDouble();

                  // increase the number of periods
                  periods++;
                  }

            // add coverage record list of a cloud to a global data
            // structure
            data.add(coverageRecordList);
            }

      // increasing the contour index to analyze the next cloud on the
      // sky image
      secondContourIndex++;
      }
}

// releasing the memory reserved by image containers
hierarchyForSky.release();
hierarchyForClouds.release();
erodeElementForSky.release();
erodeElementForClouds.release();
dilateElementForClouds.release();
morphOutputForSky.release();
morphOutputForClouds.release();
maskForSun.release();
maskForClouds.release();
hsvSkyImage.release();
hsvCloudsImage.release();
originalSkyImageMat.release();
originalCloudsImageMat.release();

// increasing the image number
```

```
        imageNumber++;

        // if twelve images already processed, break the loop
        if (imageNumber> 12) {
                break;
        }
}
}

// taking the time needed for algorithm to finish
longend = System.currentTimeMillis();

// printing the information to the console
System.out.println("Algorithm lasted = " + (end - start));

// preparing to store the collected data to the CSV file
StringBuilder csvString = new StringBuilder("Time,");

for (int i = 1; i <= data.size(); i++) {
        csvString.append(i);
        if (i <= data.size()) {
                csvString.append(",");
        }
}

// adding column data for the CSV file
csvString.append("Sum");
csvString.append("\n");

// cloud counter initialization
int maxNumberOfCloudMovement = 0;

// getting the information about the cloud with the maximum number of
// movements to adjust the number of clouds for the CSV file
for (List<CoverageRecord> coverageRecordList : data) {
        if (coverageRecordList.size() > maxNumberOfCloudMovement) {
                maxNumberOfCloudMovement = coverageRecordList.size();
        }
}

// creating 2D array for CSV file structure
String[][] dataArray = new String[data.size() + 1][maxNumberOfCloudMovement];

// setting the coverage data to the 2D array
for (int i = 0; i <data.size(); i++) {
        for (int j = 0; j <data.get(i).size(); j++) {
                if (dataArray[0][j] == null) {
                        dataArray[0][j] =
                                data.get(i).get(j).getTime().format(
                                        DateTimeFormatter.ISO_TIME);
                }
                dataArray[i + 1][j] =
                        String.valueOf(
                                data.get(i).get(j).getCoveragePercentage());
        }
}
```

```
// storing coverage data to the Stringbuffer that will be
// persisted to the CSV file
for (int i = 0; i <maxNumberOfCloudMovement; i++) {
	doublecoverageSum = 0;
	for (int j = 0; j <data.size() + 1; j++) {
		csvString.append(dataArray[j][i]);
		if (j > 0 &&dataArray[j][i] != null) {
			coverageSum += Double.valueOf(dataArray[j][i]);
		}
		if (j <= data.size()) {
			csvString.append(",");
		}
	}
	csvString.append(coverageSum);
	csvString.append("\n");
}

// storing data to the CSV file
PrintWriter csvPw = new PrintWriter(new File("data.csv"));
csvPw.print(csvString);
csvPw.flush();
csvPw.close();
}

/**
* Returns the distance between two points.
	 *
	 * @paGBfirstPoint first point coordinates
	 * @param secondPoint second point coordinates
	 * @return distance between the first and the second point
	 */
	private static double calculateDistance(Point firstPoint,
		Point secondPoint) {

		double differenceX = firstPoint.x - secondPoint.x;
		double differenceY = firstPoint.y - secondPoint.y;

		double distance = Math.sqrt(Math.pow(differenceX, 2)
			+ Math.pow(differenceY, 2));

		return distance;
	}

	/**
	 * It contains data related to sun coverage record: ordinal number of
	* the cloud contour on the image, time and coverage percentage.
	 *
	 * @author Aleksander
	 *
	 */
	static class CoverageRecord {
		private int cloudOrdinal;
		private LocalTime time;
		private double coveragePercentage;

		/**
		 * Creates the coverage record object.
```

```
	 *
	 * @paramcloudOrdinal-ordinal number of the cloud
	*      contour on the sky image
	 * @param time               - time of the day when the sun
	*                              coverage is recorded
	 * @param coveragePercentage-percentage of the sun area
	*                              covered by clouds
	 */
	public CoverageRecord(int cloudOrdinal, LocalTime time,
		double coveragePercentage)
	{
		super();
		this.cloudOrdinal = cloudOrdinal;
		this.time = time;
		this.coveragePercentage = coveragePercentage;
	}

	public double getCoveragePercentage() {
		return coveragePercentage;
	}

	public void setCoveragePercentage(double coveragePercentage) {
		this.coveragePercentage = coveragePercentage;
	}

	publicint getCloudOrdinal() {
		return cloudOrdinal;
	}

	public void setCloudOrdinal(int cloudOrdinal) {
		this.cloudOrdinal = cloudOrdinal;
	}

	public LocalTime getTime() {
		return time;
	}

	public void setTime(LocalTime time) {
		this.time = time;
	}
	}
}
```

Prilog D. Skripta za traženje graničnog iznosa energije za bat

```
%kreiranje strukture za optimizaciju
opt=optimset;
%postavljanje ispisa informacija nakon završetka iteracije
opt=optimset(opt,'Display','iter');
%postavljanje maksimalnog broja iteracija
opt=optimset(opt,'MaxIter',50);
%traženje minimalne granične vrijednosti za energiju baterije korištenjem
%funkcije 'korisnost' (Prilog E) s početnim uvjetom postavljenim na '0'
gr=fminsearch('korisnost',0,opt);
```

Prilog E. Primjer ispisa procesa optimiranja iz Matlaba

```
function f=korisnost(u)
%funkcija inverzne korisnosti čiji se minimum traži
%u - donja granica napunjenosti baterije kad se uključuje vodik

set_param('Sustav_pohrane/E_bat_gr','Value',mat2str(u));
sim('Sustav_pohrane');

%Ep - vektor energija [E_SC E_BAT E_h2]
%P  - vektor snaga [Pu PsrPp]
%P_ref - vektor ref snaga pohrane [P_SC_refP_BAT_ref P_h2_ref]
%P_p - snaga pohrane [P_SC P_bat P_h2]
%I_Pp - integral ukupne snage pohrane
%I_Pu - integral ukupne proizvedene snage

%vektor korisnosti pražnjenja [eta_SCeta_bat eta_h2]
eta=[sqrt(0.95) sqrt(0.8) 0.5];

%inverzna korisnost
f=(I_Pp(end)+sum((Ep(1,:)-Ep(end,:))./eta));
```

Prilog F. Primjer ispisa procesa optimiranja iz Matlaba

```
>> optimiranje

 Iteration  Func-count     min f(x)         Procedure
     0          1            18.4876
     1          2            18.478       initialsimplex
     2          4            18.4503        expand
     3          6            18.4027        expand
     4          8            18.3005        expand
     5          10           18.0997        expand
     6          12           17.6985        expand
     7          14           16.9         expand
     8          16           15.2952        expand
     9          18           12.089       expand
```

10	20	8.3468	expand
11	22	8.3468	contractoutside
12	24	8.3468	contractinside
13	26	8.26802	reflect
14	28	8.26802	contractinside
15	30	8.22875	reflect
16	32	8.22875	contractinside
17	34	8.22875	contractinside
18	36	8.22875	contractinside
19	38	8.22375	reflect
20	40	8.22375	contractinside
21	42	8.22375	contractinside
22	44	8.22259	reflect
23	46	8.22259	contractinside
24	48	8.22191	reflect
25	50	8.22191	contractinside
26	52	8.22191	contractinside
27	54	8.22188	contractinside

Optimizationterminated:
thecurrent x satisfiestheterminationcriteriausingOPTIONS.TolXof 1.000000e-04
and F(X) satisfiestheconvergencecriteriausingOPTIONS.TolFunof 1.000000e-04

Elapsed time is 141.205183 seconds.
>>

Životopis

Aleksander Radovan je rođen 1981. godine u Ljubljani, Republika Slovenija. Završio je srednju školu Tehnička i obrtnička škola u Čakovcu 1999. godine nakon čega je iste godine upisao Fakultet elektrotehnike i računarstva u Zagrebu, na kojem diplomira računarstvo 2004. godine pod mentorstvom Lea Budina. U veljači 2012. upisuje doktorski studij. Kvalifikacijski doktorski ispit polaže 2014. godine, a javni razgovor 2018. godine.

Još tijekom studija od 2003. godine počinje raditi kao Java programer u tvrtki Nove Tehnologije, a od 2016. u tvrtki CROZ. Također honorarno od 2005. godine radi kao asistent, a kasnije i kao predavač predmetima vezanim uz programiranje i razvoj informacijskih sustava na sljedećim visokim učilištima: Tehničko veleučilište u Zagrebu, Veleučilište Velika Gorica, Visoko učilište Algebra i Rochester Institute of Technology Croatia. Od 2014. godine počinje raditi kao voditelj razvojnog tima u tvrtki King ICT, a od 2019. godine u tvrtki BISS d.o.o. kao direktor razvoja. Od 2013. godine je aktivan i u udruzi Hrvatska udruga Java korisnika.

Popis radova:

Obrazovni materijali

Priručnik

Radovan, Aleksander; Mihaljević, Branko
Interoperabilnost informacijskih sustava, 3. izmijenjeno i dopunjeno izdanje.
Zagreb: Algebra, 2020

Kučak, Danijel; Radovan, Aleksander
Programiranje u Javi II. / Ćosić Barbara (ur.).
Zagreb: Algebra d.o.o., 2019

Radovi u časopisima

Znanstveni i pregledni radovi

Radovan, Aleksander; Šunde, Viktor; Kučak, Danijel; Ban, Željko
Solar Irradiance Forecast Based on Cloud Movement Prediction. // Energies, Energies 14 (2021), 13; 3775, 26 doi:10.3390/en14133775 (međunarodna recenzija, članak, znanstveni)

Radovi u zbornicima skupova

Znanstveni radovi u zbornicima skupova

Šipek, Matija; Muharemagić, Dino; Mihaljević, Branko; Radovan, Aleksander
Next-Generation Web Applications with WebAssembly and TruffleWasm. // Proceedings of the 44th International Convention for Information and Communication Technology, Electronics and Microelectronics - MIPRO 2021 / Skala, Karolj (ur.).
Opatija: MIPRO, 2021. str. 1950-1955 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Beronić, Dora; Pufek, Paula; Mihaljević, Branko; Radovan, Aleksander
On Analyzing Virtual Threads – a Structured Concurrency Model for Scalable Applications on the JVM. // Proceedings of the 44th International Convention for Information and Communication Technology, Electronics and Microelectronics - MIPRO 2021 / Skala, Karolj (ur.).
Opatija: MIPRO, 2021. str. 1939-1944 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Mihaljević, Branko; Jureković, Darko; Radovan, Aleksander; Žagar, Martin; Mutka, Alan
Leveraging Learning Software Development in Java Programming Language with Support of Oracle Academy Curriculum – a Case Study in Croatia. // Proceedings of EDULEARN21 Conference
Španjolska: IATED, 2021. str. 8495-8505 doi:.org/10.21125/edulearn.2021 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Pufek, Paula; Beronić, Dora; Mihaljević, Branko; Radovan, Aleksander
Achieving Efficient Structured Concurrency through Lightweight Fibers in Java Virtual Machine. // Proceedings of the 43rd International Convention for Information and Communication Technology, Electronics and Microelectronics (MIPRO 2020) / Skala, Karolj (ur.).
Opatija, Hrvatska: MIPRO, 2020. str. 2082-2087 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Mihaljević, Branko; Žagar, Martin; Radovan, Aleksander
On Challenges of Distance Delivery of Information Technology and Software Development Courses. // INTED2020 Proceedings
Valencija: IATED, 2020. str. 4627-4637 doi:10.21125/inted.2020.1283 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Mihaljević, Branko; Žagar, Martin; Radovan, Aleksander
Educational Services, Tools, and Infrastructure for Remote Delivery of Software Development Courses in Web and Mobile Computing. // Proceedings of EDULEARN20 Conference / Gómez Chova, L. ; López Martínez, A. ; Candel Torres, I. (ur.).
Palma de Mallorca: IATED, 2020. str. 8370-8380 doi:10.21125/edulearn.2020.2061 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Radovan, Aleksander; Mihaljević, Branko; Žagar, Martin
Experiences Related to Using Online Communication Tools for Distance Learning. // Proceedings of EDULEARN20 Conference / Gómez Chova, L. ; López Martínez, A. ; Candel Torres, I. (ur.).
Palma de Mallorca: IATED, 2020. str. 8386-8389 doi:10.21125/edulearn.2020.2063 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Šipek, Matija; Muharemagić, Dino; Mihaljević, Branko; Radovan, Aleksander
Enhancing Performance of Cloud-based Software Applications with GraalVM and Quarkus. // Proceedings of the 43rd International Convention for Information and Communication Technology, Electronics and Microelectronics (MIPRO 2020) / Skala, Karolj (ur.).
Opatija, Hrvatska: MIPRO, 2020. str. 2076-2081 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Šipek, Matija; Mihaljević, Branko; Radovan, Aleksander
Exploring Aspects of Polyglot High-Performance Virtual Machine GraalVM. // Proceedings of the 42nd International Convention for Information and Communication Technology, Electronics and Microelectronics (MIPRO 2019) / Skala, Karolj (ur.).
Rijeka: MIPRO, 2019. str. 1940-1945 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Grgić, Hrvoje; Mihaljević, Branko; Radovan, Aleksander
Comparison of Garbage Collectors in Java Programming Language. // Proceedings of the 41st International Convention for Information and Communication Technology, Electronics and Microelectronics (MIPRO 2018)
Opatija, Hrvatska, 2018. str. 1785-1790 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Bach, Leo; Mihaljević, Branko; Radovan, Aleksander
Exploring HTTP/2 Advantages and Performance Analysis using Java 9. // Proceedings of the 40th International Convention for Information and Communication Technology, Electronics and Microelectronics (MIPRO 2017) / Petar Biljanović (ur.).
Rijeka: Croatian Society for Information and Communication Technology, Electronics and Microelectronics - MIPRO, 2017. str. 1522-1527 doi:10.23919/MIPRO.2017.7973663 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Valenčić, Davorin; Danijela, Kažović; Radovan, Aleksander
OpenProj - open source softverski alat za upravljanje IT projektima. // MIPRO 2013 - 36. međunarodni skup za informacijsku i komunikacijsku tehnologiju, elektroniku i mikroelektroniku
Opatija, Hrvatska, 2013. str. 987-991. (https://www.bib.irb.hr/645631) (predavanje, domaća recenzija, cjeloviti rad (in extenso), znanstveni)

Stručni radovi u zbornicima skupova

Radovan, Aleksander; Mihaljević, Branko; Marasović, Kristina
Iskustva u korištenju antiplagijatskog softvera u nastavi. // Zbornik radova 19. CARNetove korisničke konferencije / 19th CARNet User Conference - CUC 2017
Dubrovnik, Hrvatska, 2017. str. 1-8 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Mihaljević, Branko; Radovan, Aleksander; Marasović, Kristina
Iskustva popularizacije programskog jezika Java te prateće edukacijske i ostale aktivnosti. // Zbornik radova 18. CARNetove korisničke konferencije / 18th CARNet User Conference - CUC 2016 / Blažetić, Ana (ur.).
Zagreb: Carnet, 2016. 1, 60 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Mihaljević, Branko; Marasović, Kristina; Radovan, Aleksander
Iskustva nastave na daljinu – sustavi, usluge i alati u predmetima iz područja informacijskih tehnologija i računarstva. // Zbornik radova 18. CARNetove korisničke konferencije / 18th CARNet User Conference - CUC 2016 / Blažetić, Ana (ur.).
Zagreb: Carnet, 2016. str. 50-60 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Radovan, Aleksander; Mihaljević, Branko
Prijedlog optimalnog korištenja gradiva matematike za usavršavanje vještina za programiranje. // Zbornik radova 18. CARNetove korisničke konferencije / 18th CARNet User Conference - CUC 2016 / Blažetić, Ana (ur.).
Zagreb: Carnet, 2016. 3, 70 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Žagar, Marinko; Radovan, Aleksander
Prijedlog implementacije informacijskog sustava za predikciju kriznih događaja. // Zbornik radova / dr.sc. Ivan Nađ (ur.).
Velika Gorica: Veleučilište Velika Gorica, 2014. str. 837-845 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Raić, Boris; Radovan, Aleksander
Microcontroller managed module for automatic ventilation of vehicle interior. // MIPRO Conference proceedings / Petar Biljanović (ur.).
Rijeka: Croatian Society for Information and Communication Technology, Electronics and Microelectronics - MIPRO, 2014. str. 1859-1864 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), stručni)

Radovan, Aleksander; Ban, Željko
Predictions of Cloud Movements and the Sun Cover Duration. // MIPRO Conference proceedings / Petar Biljanović (ur.).
Rijeka: Croatian Society for Information and Communication Technology, Electronics and Microelectronics - MIPRO Office: Kružna 8/II, P. O. Box 303, HR-51001 Rijeka,, 2014. str. 1460-1465 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso),

stručni)

Valenčić, Davorin; Radovan, Aleksander; Gligora, Tomislav
Primjer pristupa edukaciji za upravljanje projektima na IT visokoškolskom studiju. // Zbornik radova CUC 2012 - Brže (komunicirati), više (naučiti), jače (se povezati)! / Orlović, Ana (ur.).
Zagreb: Hrvatska akademska i istraživačka mreža - CARNet, 2013. str. 1580-1592. (https://www.bib.irb.hr/645609) (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Radovan, Aleksander; Žagar, Marinko
Korištenje naučenih lekcija u svrhu kontrole rizika i osiguravanje održivosti određenih dijelova informacijskih sustava i projekata. // Zbornik radova VI. međunarodne konferencije "Dani kriznog upravljanja"
Velika Gorica, Hrvatska, 2013. str. 1432-1446. (https://www.bib.irb.hr/968620) (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), stručni)

Matić, Ivan; Gligora, Tomislav; Radovan, Aleksander
Studentski pristup razvoju informacijskog sustava za potporu provedbe postupka javne nabave. // Zbornik radova CUC 2013 - Dohvati znanje! / Orlović, Ana (ur.).
Zagreb: Hrvatska akademska i istraživačka mreža - CARNet, 2013. str. 78-90 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Valić, Bruno; Radovan, Aleksander; Pavlović, Damir
Korištenje programskog jezika Scratch za podučavanje osnova programiranja od malih nogu. // Zbornik radova CUC 2013 - Dohvati znanje! / Orlović, Ana (ur.).
Zagreb: Hrvatska akademska i istraživačka mreža - CARNet, 2013. str. 451-461 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Žagar, Marinko; Radovan, Aleksander;
Korištenje naučenih lekcija u svrhu kontrole rizika i osiguravanje održivosti sustava i projekata. // Zbornik radova / Prof. mr.sc. Ivan Toth (ur.).
Velika Gorica: Veleučilište Velika Gorica, 2013. str. 1423-1437 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Radovan, Aleksander; Gligora, Tomislav; Valenčić, Davorin
Agilna metodologija razvoja mobilnih aplikacija za komunikaciju u kriznim situacijama. // Zbornik radova / Prof. mr.sc. Ivan Toth (ur.).
Velika Gorica: Veleučilište Velika Gorica, 2013. str. 1376-1395 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), stručni)

Božičević, Mislav; Radovan, Aleksander
A HTTP-Enabled Cryptographically Secure Hardware Random Number Generator. // MIPRO 2013
Opatija, Hrvatska, 2013. str. 1234-1242 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), stručni)

Radovan, Aleksander; Gligora, Tomislav; Valenčić, Davorin
Prijedlog podučavanja agilnih metoda razvoja softvera korištenjem alata Microsoft Project 2010. // Zbornik radova CUC 2012 - Brže (komunicirati), više (naučiti), jače (se povezati)! / Orlović, Ana (ur.).
Zagreb: Hrvatska akademska i istraživačka mreža - CARNet, 2013. str. 89-100 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Gligora, Tomislav; Valenčić, Davorin; Radovan, Aleksander
Implementacija metodologije ekstremnog programiranja u nastavni proces visokoobrazovnih institucija. // Zbornik radova CUC 2012 - Brže (komunicirati), više (naučiti), jače (se povezati)! / Orlović, Ana (ur.).
Zagreb: Hrvatska akademska i istraživačka mreža - CARNet, 2013. str. 56-68 (predavanje, domaća recenzija, cjeloviti rad (in extenso), stručni)

Drugi radovi u zbornicima skupova

Draženović, Karla; Radovan, Aleksander; Šunde, Viktor; Ban, Željko
Optimiranje hibridnog sustava za pohranu električne energije. // 14. savjetovanje HRO CIGRÉ / FILIPOVIĆ-GRČIĆ, BOŽIDAR (ur.).
Zagreb: HRO CIGRE, 2019. str. 1-14. (https://www.bib.irb.hr/1041027) (predavanje, domaća recenzija, cjeloviti rad (in extenso), ostalo)

Sažeci sa skupova

Sažeci u zbornicima i časopisima

Iveta, Mateja; Radovan, Aleksander; Mihaljević, Branko;
Prediction of Traffic Accidents Severity Based on Machine Learning and Multiclass Classification Model. // Proceedings of the 44th International Convention for Information and Communication Technology, Electronics and Microelectronics - MIPRO 2021 / Skala, Karolj (ur.).
Opatija: MIPRO, 2021. str. 1956-1960 (predavanje, međunarodna recenzija, sažetak, znanstveni)

Mihaljević, Branko; Radovan, Aleksander; Žagar, Martin; Matijašević, Stjepan
The State of Java - Today and Tomorrow. // 7th International Java Community Conference in Croatia - Javantura v7
Zagreb: HUJAK, 2020. 1, 73 (plenarno, recenziran, sažetak, stručni)

Radovan, Aleksander
Tehnike sigurnog programiranja. // II. znanstvena i stručna konferencija "Kibernetička sigurnost i digitalna forenzika"
Zagreb, 2019. str. 60-65 (pozvano predavanje, domaća recenzija, sažetak, stručni)

Mihaljević, Branko; Radovan, Aleksander
Java in Croatia and HUJAK. // 6th International Community Java Conference in

Croatia - Javantura v6
Zagreb: HUJAK, 2019. 1, 38 (plenarno, recenziran, sažetak, stručni)

Radovan, Aleksander; Mihaljević, Branko
Controlling solar electric system using Java. // 8th International Java Conference in Croatia - JavaCro'19
Umag, Hrvatska, 2019. 1, 38 (predavanje, međunarodna recenzija, sažetak, stručni)

Mihaljević, Branko; Radovan, Aleksander; Žagar , Martin
Software Development in Java - Today and Tomorrow. // 24th Croatian Oracle User Group Conference - HrOUG 2019
Rovinj, Hrvatska, 2019. str. 1-80 (pozvano predavanje, međunarodna recenzija, sažetak, stručni)

Radovan, Aleksander; Mihaljević, Branko
Java and Artificial Intelligence – tools and libraries. // 7th International Java Conference in Croatia - JavaCro'18
Rovinj, Hrvatska, 2018. str. 1-20 (predavanje, recenziran, sažetak, stručni)

Radovan, Aleksander; Mihaljević, Branko
Immutable collections in Java 9 – do we need them?. // DevTalks Bucharest 2018
Bukurešt, Rumunjska, 2018. str. 1-20 (predavanje, međunarodna recenzija, sažetak, stručni)

Radovan, Aleksander; Ban, Željko
Prediction of HSV color model parameter values of cloud movement picture based on artificial neural networks. // Zbornik radova 41. međunarodnog skupa za informacijsku i komunikacijsku tehnologiju, elektroniku i mikroelektroniku
Opatija, Hrvatska, 2018. str. 1110-1114 (predavanje, međunarodna recenzija, sažetak, stručni)

Radovan, Aleksander; Mihaljević, Branko
Immutable collections in Java 9 – do we need them?. // 5th International Community Java Conference in Croatia - Javantura v5
Zagreb, Hrvatska, 2018. str. 1-20 (predavanje, recenziran, sažetak, stručni)

Mihaljević, Branko; Žagar, Martin; Radovan, Aleksander
Java on the Blockchain. // 7th International Java Conference in Croatia - JavaCro'18
Rovinj, Hrvatska, 2018. str. 1-30 (predavanje, međunarodna recenzija, sažetak, stručni)

Radovan, Aleksander; Mihaljević, Branko
Java lambdas and streams – are they better than for loops?. // 4th International Community Java Conference in Croatia - Javantura v4
Zagreb, Hrvatska, 2017. str. 1-28 (predavanje, međunarodna recenzija, sažetak, stručni)

Mihaljević, Branko; Radovan, Aleksander
The Way of Java Certification. // 21st Croatian Oracle User Group Conference - HrOUG 2016

Rovinj, Hrvatska, 2016. str. 70-77 (pozvano predavanje, recenziran, sažetak, stručni)

Mihaljević, Branko; Radovan, Aleksander
HUJAKization of Java Community. // 21st Croatian Oracle User Group Conference - HrOUG 2016
Rovinj, Hrvatska, 2016. str. 48-60 (pozvano predavanje, recenziran, sažetak, stručni)

Mihaljević, Branko; Smuđ, Tihomir; Radovan, Aleksander
Java Certification – in theory and practice. // 4th International Java Conference in Croatia - JavaCro'15
Rovinj, Hrvatska, 2015. str. 80-88 (pozvano predavanje, recenziran, sažetak, stručni)

Mihaljević, Branko; Radovan, Aleksander
Svijet Jave - u Hrvatskoj. // Zbornik međunarodne konferencije JavaCro'14 konferencije / Radić, Vladimir (ur.).
Zagreb: Printera grupa, 2014. str. 23-23 (pozvano predavanje, međunarodna recenzija, sažetak, stručni)

Ćutić, Juraj; Radovan, Aleksander
Automatizirano testiranje sa Seleniumom 2. // Zbornik međunarodne konferencije JavaCro'14 konferencije / Vladimir Radić (ur.).
Zagreb: Printera grupa, 2014. str. 45-45 (predavanje, sažetak, stručni)

Radovan, Aleksander; Kramberger, Tin; Gligora Tomislav
Od igre s kockicama do programera – Lego Mindstorms. // CUC 2014
Zagreb: Carnet, 2014. str. 87-95. (https://www.bib.irb.hr/891253) (predavanje, domaća recenzija, sažetak, ostalo)

Ćutić, Juraj; Radovan, Aleksander
Java library za implementaciju programskih rješenja za fiskalizaciju u RH. // Zbornik međunarodne konferencije JavaCro'13 konferencije / Vladimi Radić (ur.).
Zagreb: PRINTERA GRUPA, 2013. str. 33-33 (predavanje, sažetak, ostalo)

Mihaljević, Branko; Radovan, Aleksander
Proširimo vidike - kako osloboditi naš potencijal u Javi. // Zbornik 18. konferencije Hrvatske udruge Oracle korisnika - HrOUG 2013 / Radić, Vladimir (ur.).
Rovinj: HrOUG, 2013. str. 87-87 (plenarno, recenziran, sažetak, stručni)

Mihaljević, Branko; Radovan, Aleksander
U iščekivanju Java SE 8. // Zbornik 18. konferencije Hrvatske udruge Oracle korisnika / Radić, Vladimir (ur.).
Zagreb: HrOUG, 2013. str. 44-46 (predavanje, domaća recenzija, sažetak, stručni)

Radovan, Aleksander; Gligora, Tomislav
Razvoj Java web aplikacija korištenjem Wavemaker razvojnog okruženja. // Zbornik 17. konferencije Hrvatske udruge Oracle korisnika / Vladimi Radić (ur.).
Sv.Nedjeljka: PRINTERA GRUPA, 2012. str. 103-103 (predavanje, sažetak, ostalo)

Druga sudjelovanja na skupovima

Mihaljević, Branko; Žagar, Martin; Radovan, Aleksander; Matijašević, Stjepan
Java is Here to Stay. // Proceedings of the 9th International Java Conference in Croatia - JavaCro'21
Zagreb, 2021. (plenarno, recenziran, pp prezentacija, stručni)

Mihaljević, Branko; Radovan, Aleksander; Žagar, Martin
The Best Java Tools for Developers in 2021. // Proceedings of the 9th International Java Conference in Croatia - JavaCro'21
Zagreb, 2021. (predavanje, recenziran, pp prezentacija, stručni)

Radovan, Aleksander; Hoxha, Vjori; Barić, Ivor
Predicting future using Java and neural networks. // Javantura v6
Zagreb, Hrvatska, 2019. (predavanje, domaća recenzija, pp prezentacija, stručni)

Mihaljević, Branko; Radovan, Aleksander; Žagar, Martin
The State of Java and Software Development in Croatia. // 8th International Java Conference in Croatia - JavaCro'19
Umag, Hrvatska, 2019. str. 1-69 (plenarno, recenziran, pp prezentacija, stručni)

Radovan, Aleksander
Is Java still free?. // Change conference 2018.
Zagreb, Hrvatska, 2018. (predavanje, međunarodna recenzija, neobjavljeni rad, stručni)

Mihaljević, Branko; Radovan, Aleksander
The (Smart) World of Java – Revisited. // 7th International Java Conference in Croatia - JavaCro'18
Rovinj, Hrvatska, 2018. (plenarno, recenziran, pp prezentacija, stručni)

Mihaljević, Branko; Radovan, Aleksander
CroDuke Indy and the Temple of Java Boom. // 5th International Community Java Conference in Croatia - Javantura v5
Zagreb, Hrvatska, 2018. (plenarno, recenziran, pp prezentacija, stručni)

Mihaljević, Branko; Radovan, Aleksander
Java - One Thing to Rule Them (Us) All. // 6th International Java Conference in Croatia - JavaCro'17
Rovinj, Hrvatska, 2017. (plenarno, međunarodna recenzija, pp prezentacija, stručni)

Mihaljević, Branko; Radovan, Aleksander;
CroDuke Indy and the Kingdom of Java Skills. // 4th International Community Java Conference in Croatia - Javantura v4
Zagreb, Hrvatska, 2017. (plenarno, međunarodna recenzija, pp prezentacija, stručni)

Dujmović, Matija; Radovan, Aleksander
True RESTful Java web services with Elide and Katharsis. // Javantura 2017

Zagreb, Hrvatska, 2017. (predavanje, međunarodna recenzija, pp prezentacija, stručni)

Radovan, Aleksander
Java 8 and Lambdas - did we expect too much?. // Change conference 2016
Zagreb, Hrvatska, 2016. (predavanje, domaća recenzija, pp prezentacija, stručni)

Mitar, Danijel; Radovan, Aleksander
JVM problem diagnostics. // JavaCro 2016
Rovinj, 2016. (predavanje, međunarodna recenzija, pp prezentacija, stručni)

Mihaljević, Branko; Radovan, Aleksander
(Why) We Live in the World of Java. // 5th International Java Conference in Croatia - JavaCro'16
Rovinj, Hrvatska, 2016. (plenarno, recenziran, pp prezentacija, stručni)

Mihaljević, Branko; Matijašević, Stjepan, Đurđević, Hrvoje; Žnidarić, Slavko; Orlić, Marin; Radovan, Aleksander
The Story of Java & HUJAK. // 3rd International Community Java Conference in Croatia - Javantura v3
Zagreb, Hrvatska, 2016. (plenarno, recenziran, pp prezentacija, stručni)

Mihaljević, Branko; Radovan, Aleksander
HUJAKing – Expansion of Java Community. // 4th International Java Conference in Croatia - JavaCro'15
Rovinj, Hrvatska, 2015. (plenarno, recenziran, pp prezentacija, stručni)

Radovan, Aleksander; Ćutić, Juraj
Automatizirano testiranje sa Seleniumom 2. // JavaCro '14 - 3rd International Java Conference in Croatia
Poreč, 2014. (predavanje, međunarodna recenzija, pp prezentacija, ostalo)

Mihaljević, Branko; Radovan, Aleksander; Vukmanović, Duško
The Road to Java. // 2nd International Community Java Conference in Croatia - Javantura v2
Zagreb, Hrvatska, 2014. (plenarno, recenziran, pp prezentacija, stručni)

Ćutić, Juraj; Radovan, Aleksander
Je li JavaFX pravi nasljednik Swinga?. // Javantura 2014
Zagreb, 2014. str. 60-66 (predavanje, sažetak, stručni)

Mihaljević, Branko; Radovan, Aleksander
The World of Java - in Croatia. // 3rd International Java Conference in Croatia – JavaCro'14
Poreč, Hrvatska, 2014. (plenarno, recenziran, pp prezentacija, stručni)

Mihaljević, Branko; Matijašević, Stjepan; Đurđević, Hrvoje; Žnidarić, Slavko; Orlić, Marin; Radovan, Aleksander; Križanić, Jurica
Three Years of Happiness. // 2nd Community Java Conference in Croatia - Javantura

v2
Zagreb, Hrvatska, 2014. (plenarno, recenziran, pp prezentacija, stručni)

Ostale vrste radova

Mihaljević, Branko; Radovan, Aleksander
Proljetna Java 10., 2018. (recenziran, popularan rad).

Mihaljević, Branko; Radovan, Aleksander
Modul do modula - nova Java 9 slagalica., 2017. (recenziran, popularan rad).

Curriculum vitae

Aleksander Radovan was born in 1981 in Ljubljana, Republic of Slovenia. He graduated from the Tehnička i obrtnička škola in Čakovec in 1999, after which he enrolled at the Faculty of Electrical Engineering and Computing in Zagreb in the same year, where he graduated in computer science in 2004 under the mentorship of Leo Budin. In February 2012, he enrolled in doctoral studies. He takes the qualifying doctoral exam in 2014, and the public interview in 2018.

During his studies, he started working as a Java programmer in the company Nove Tehnologije in 2003, and in 2016 in the company CROZ. He has also been workingp art-time since 2005 as an assistant, and later as a lecturer in subjects related to programming and development of in formation systems at the following universities: University of Applied Sciences Zagreb, University of Applied Sciences Velika Gorica, Algebra University College and Rochester Institute of Technology Croatia. From 2014 he started working as a development team leader at King ICT, and from 2019 at BISS d.o.o. as director of development. Since 2013, he has been active in the Croatian Java User Group.