Milenko Gavrić

Prepoznavanje meteora sa serije uzastopnih snimaka all-sky kamere

Razvijena je aplikacija METeorDEteCTor (METDECT) koja izdvaja slike koje potencijalno sadrže meteor(e) iz serije snimaka napravljenih all-sky kamerom sa fish-eye objektivom. Analizirani su radovi koji se bave analizom i obradom slika slikanih kamerom sa fish-eye objektivom i detaljno su analizirani njihovi algoritmi koji su delom i primenjeni, uz odgovarajuća prilagođavanja. Definisan je algoritam kojim se izdvajaju slike sa mogućim prisustvom meteora. Objašnjene su korišćene klase i njihovi atributi, kao i metode razvijene tokom implementacije definisanog algoritma. Razvijena je biblioteka u C# za podršku implementacije definisanog algoritma. Definisana je klasa koja kao metode sadrži opšte algoritme za obrađivanje slika korišćenih u ovom projektu. Glavni algoritmi koji su korišćeni u toj klasi su algoritam za generisanje Bezijerove krive, algoritam za pronalaženje granične vrednosti (treshold), algoritam za pretragu u širinu, algoritam za pronalaženje prave najbliže datim tačkama i dilatacija. Zasebna klasa je namenjena za čuvanje izdvojenih objekata (zvezde i mogući meteori), kao i njihovo dalje proveravanje. Takođe je korišćena i biblioteka Aforge.Imaging u cilju definisanja željene palete boja. Prikazani su dobijeni rezultati analize 3390 slika napravljenih petničkom kamerom i opisani potencijalni smerovi budućeg razvoja.

Uvod

Cilj ovog projekta je da se izdvoje slike snimljene all-sky kamerom koje potencijalno sadrže meteore od slika bez njih. Nije dozvoljeno da se izbaci slika koja sadrži meteor, po cenu da se ostave i neke slike bez meteora. Na taj način bi se pomoglo da se u mnoštvu snimaka pronađu retki koji sadrže meteor. Snimke koji su korišćeni u radu snimila je Petnička kamera za bolide, odnosno veoma sjajne meteore, POgLED (Petnica bOlide alL-sky camEra Device, Bettonvil 2015). Tokom izrade projekta razvijena je biblioteka u programskom jeziku C#, koja sadrži neophodne klase i metode za obradu slika. Dizajnirana je i implementirana aplikacija u programskom jeziku C# koja koristi razvijenu biblioteku i kao izlaz izdvaja slike koje potencijalno sadrže meteor. Algoritam koji je razvijen donekle se poklapa sa algoritmom koji je razvijen u referentnim radovima uz odgovarajuće dopune i prilagođavanja. García (2008) implementira namensku funkciju koja čisti pojedinačnu sliku od svetlosnog zagađenja, drveća, meseca i oblaka, ali ovo rešenje je prilagođeno snimcima sa kratkom ekspozicijom, te nije bilo primenljivo za ovaj projekat. Reffet i saradnici (2014) se bave putanjama kojima se meteor kreće. Koriste Houghov algoritam koji detektuje prave linije. Ovo rešenje nije primenljivo u našem slučaju zbog prekidâ na tragu meteora dobijenih pomoću čopera od tečnog kristala (trag je isprekidan radi određivanja brzine meteora). Evich (2012) u svom radu analizira povećanja intenziteta piksela na susednim frejmovima. Ako detektovano povećanje traje kraće od 4 frejma ili duže od 15 sekundi, taj deo se odbacuje. Razlika između situacije u navedenom radu i naše je u tome što su u našem slučaju slike eksponirane, a ne niz frejmova. Događaj od interesa traje najviše 15 sekundi, dok ekspozicija snimaka kojima se bavi naš rad iznosi po 3 minuta.

Milenko Gavrić (1999), Novi Sad, Jirečekova 7, učenik 2. razreda Gimnazije „Jovan Jovanović Zmaj" u Novom Sadu

MENTORI:

Dragan Toroman, ISP

Ivan Glišin, CPU d. o. o, Beograd

Teorijska osnova

Slike koje su obrađivane u radu su koristile RGB i HSV način zapisivanja.

Pri obradi slika korišćen je grayscale način zapisivanja. Ovaj način zapisivanja je korišćen za detekciju objekata na slici, kao i za efikasniju obradu. Za prebacivanje iz RGB prikaza u grayscale se koristila sledeća formula (Web 1):

$$G = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B \quad (1)$$

Bezijerova kriva je iskorišćena za podešavanje vrednosti koje će biti dodeljene pikselima. Na Bezijerovoj krivoj su postavljene tačke tako da se što više zvezda jasno izdvoji. Na osnovu iskustava stečenih tokom projekta, Bezijerova kriva je pravljena za plavi kanal iz RGB-a.

Slika se inicijalno konvertuje u grayscale format. Na osnovu definisanog algoritma se izračunava granična vrednost i svaki piksel čija je vrednost iznad te granice se postavlja na maksimalnu vrednost koju grayscale piksel može da ima, dok se ako ima manju vrednost od granične postavlja na minimalnu. Konačni rezultat obrade slike u aplikaciji METeorDEteCTor je crno-bela slika.

Dizajn rešenja

Inicijalni pokušaji

Da bi se došlo do prikazanog rešenja, isprobano je više ideja i algoritama, od kojih su se neki prikazali kao neprimenljivi na rešavani problem. Ovde će se diskutovati o tim pokušajima – zašto su pokušani i zašto nisu uspeli.

Za inicijalnu analizu mogućnosti pri obradi slika se koristila biblioteka FIP (Fundamental image processing methods) koju je napisao Jakub Szymanowski (Web 2). Pokazalo se da biblioteka ne sadrži potrebne funkcije, npr. flood fill, dilatation, crtanje kruga i da se sporo izvršava. Biblioteci je bilo potrebno 3 minuta da pristupi svim pikselima slike i izvrši identičnu operaciju nad svakim pikselom. Ovo je posledica neoptimalnog pristupanja pikselima slike (korišćene su ugrađene metode GetPixel i SetPixel koje zahtevaju zaključavanje i otključavanje bitmape pri svakom pristupu pikselu).

Drugi pokušaj je bilo generisanje maske samo za idealan slučaj, tj. u slučaju kada je nebo vedro. U masku spada drveće, zgrade, svetlost od električnih uređaja, oblaci, kiša, itd. Maska se generisala samo jednom, na početku izvršavanja aplikacije. U slučaju kada generisana maska ne bi odgovarala korisniku, korisnik bi mogao sam da iscrta koju površinu bi želeo da program obrađuje. Ovi pristupi su se pokazali kao neupotrebljivi, zbog velike promenljivosti na snimcima tokom oblačnog vremena. Zbog toga se implementirao algoritam koji bi generisao zasebnu masku za svaku sliku.

Pokušan je i pristup da se simulira rotiranje zvezda oko nebeskog pola. Iz ovog razmišljanja sledi da je meteor objekat koji nema isti centar kao većina zvezda. Ovakav pristup mogao bi da se primeni bilo gde na svetu, iako se menjaju koordinate centra oko kojeg se zvezde okreću. Implementacija ovog načina razmišljanja je veoma teška, zbog toga što se pokazalo da centar nije jedna tačka već površina, kao i da je ta površina promenljiva, te se odustalo od ovog pristupa.

Treći pokušaj je bio da se koristi metoda dilatacije biblioteke Aforge.Imaging. Ova metoda je potrebna da bi se odstranio šum koji je u neposrednoj blizini ivice maske, a predstavlja proširenje pojedinog piksela istom bojom u svim smerovima. Razlog zbog kojeg je metoda dilatacije u Aforge.Imaging bila spora je zato što je on za svaki piksel gledala susedne piksele i od njih birala onu vrednost koja je najveća. Dilatacija je implementirana tako da se svaki piksel proširuje samo jednom. Zbog veličine slike i promenljivosti vremenskih prilika, ova operacija je na osnovu testova trebalo da se pozove 50 puta, ali je taj parametar konfigurabilan u finalnoj implementaciji. Izvršavanje ove operacije nad celom slikom je trajalo oko 1 minut, na testnom hardveru, što je u poređenju sa gornjom granicom od 3 minuta za obradu cele slike predugo. Da bi se izvršenje aplikacije ubrzalo, implementirana je nova funkcija dilatacije.

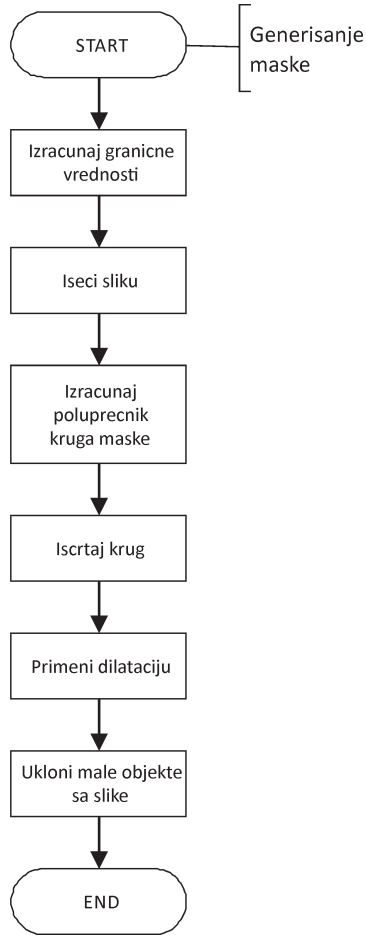
Budući da je slika crno-bela i da se proširuju samo beli pikseli, ovakva operacija nije bila potrebna. Zato je implementirana funkcija koja uvek za vrednost na koju postavlja susedne piksele ima 255, tj. belu boju. Ovo jeste ubrzalo izvršavanje funkcije, ali je ono i dalje bilo presporo.

Razmišljanje koje je dovelo do konačnog rešenja je da ako se, na primer, 50 puta primeni dilatacija na jedan beli piksel, kreiraće se kvadrat 50×50 piksela. Implementirana funkcija uzima referencu slike, kao i konfigurabilan broj puta ponavljanja. Generiše se kvadrat $N \times N$ i za svaki beli piksel pored kog se u neposrednoj blizini nalazi crni iscrtava se kvadrat koristeći Graphics.DrawImage. Ovo je višestruko ubrzalo program i na taj način omogućilo da se od 1 minuta za samo izvršavanje funkcije pređe na ispod 40 sekundi za generisanje maske.

Opis algoritma

Rešenje do kog se na kraju došlo i koje je i implementirano je da se prvo generiše niz odgovarajućih vrednosti preko Bezijerove krive za svaku vrednost plave komponente RGB-a slike koja se obrađuje. Do vrednosti koordinata za generisanje ove prave došlo se eksperimentalno, na osnovu manualnog testiranja. Zatim se plava komponenta svakog piksela u slici promeni na odgovarajuću vrednost u nizu. Plava komponenta se povećava da bi se izdvojile zvezde. U slučaju da se RGB piksel pretvara u grayscale piksel, plava komponenta ima najmanji koeficijent koji se množi, kao što pokazuje jednačina (1). Zbog toga se plava komponenta svakog objekta koja ima visoku vrednost postavlja na maksimalnu vrednost, tj. 255. Prilikom pretvaranja u grayscale format koristi se vrednost HSV-skale. Ta vrednost jednaka je maksimumu od tri RGB komponente. Na ovaj način se izdvaja veliki broj zvezda, što je bitno za nastavak algoritma.

Posle toga se generiše maska. Maska se generiše za svaku sliku, ali tako da njene dimenzije budu jednake manjoj veličini od dužine i širine slike. Tokom inicijalnog prolaska kroz sliku generiše se histogram te slike. Granična vrednost, koja će se koristiti za binarizaciju, jednaka je vrednosti u kojoj je površina između levog i desnog dela histograma jednaka. Nakon što se ova vrednost izračuna generiše se nova slika koja je crno bela. Zatim se obrišu svi objekti koji su manji od 200 piksela, što je konfigurabilno. Na ovaj način na snimcima ostaju samo zgrade, drveće i mesec. Nakon toga se računa poluprečnik kruga koji definiše površinu za analizu, a čiji je centar identičan centru slike. Poluprečnik kruga se dobija na dva načina, a uzima se manja od dve dobijene vrednosti. Prvi način polazi od pretpostavke da krug ne postoji, tj. da nema oblaka koji jasno definiše površinu za analizu. Za svaki objekat koji je ostao na slici, računa se udaljenost njegove najudaljenije tačke od centra slike i kao rezultujući poluprečnik se uzima prosek svih poluprečnika. Drugi način pretpostavlja da je oblačno, ali da je centar slike čist. Mana ovog pristupa je što se prekida analiza date slike zato što je centar slike neobradiv, odnosno pokriven nekim objektom. Nalaze se svi crni pikseli koji su okruženi belim pikselima i računa se prosečan poluprečnik. Nakon što je poluprečnik izračunat, crta se krug sa zadatim poluprečnikom, sa cen-



Slika 1. Algoritam za generisanje maske

Figure 1. Algorithm for mask generating

Slika 2. Obrađena četiri uzastopna snimka – A, B, C i D, gde se na B nalazi meteor

Figure 2. Four consecutive processed images – A, B, C and D, where B contains a meteor

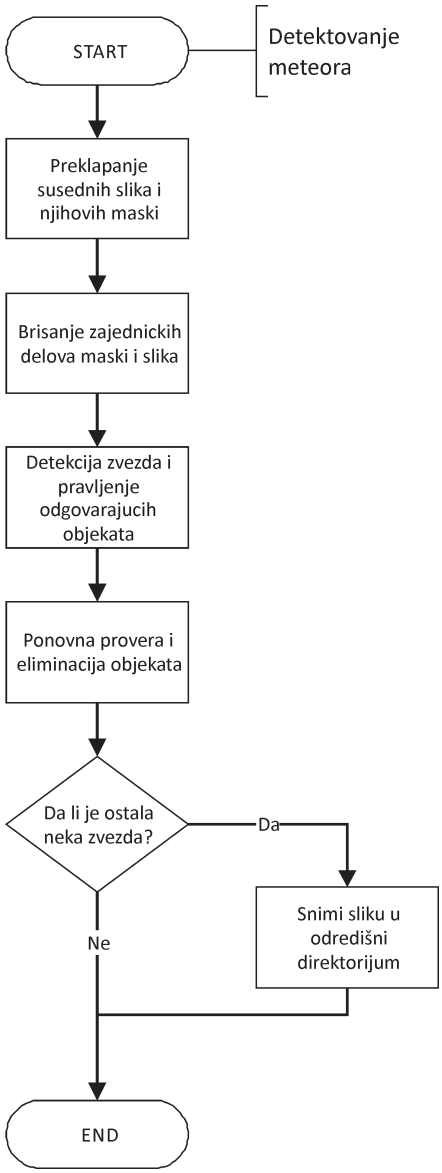
Slika 3. Detektovane razlike između uzastopnih snimaka (A-B, B-C, C-D)

Figure 3. Detected differences between conscutive images (A-B, B-C, C-D)

Slika 4. Uvećani deo snimka na kome se vidi meteor, prisutan samo na A-B i B-C (nema ga na C-D)

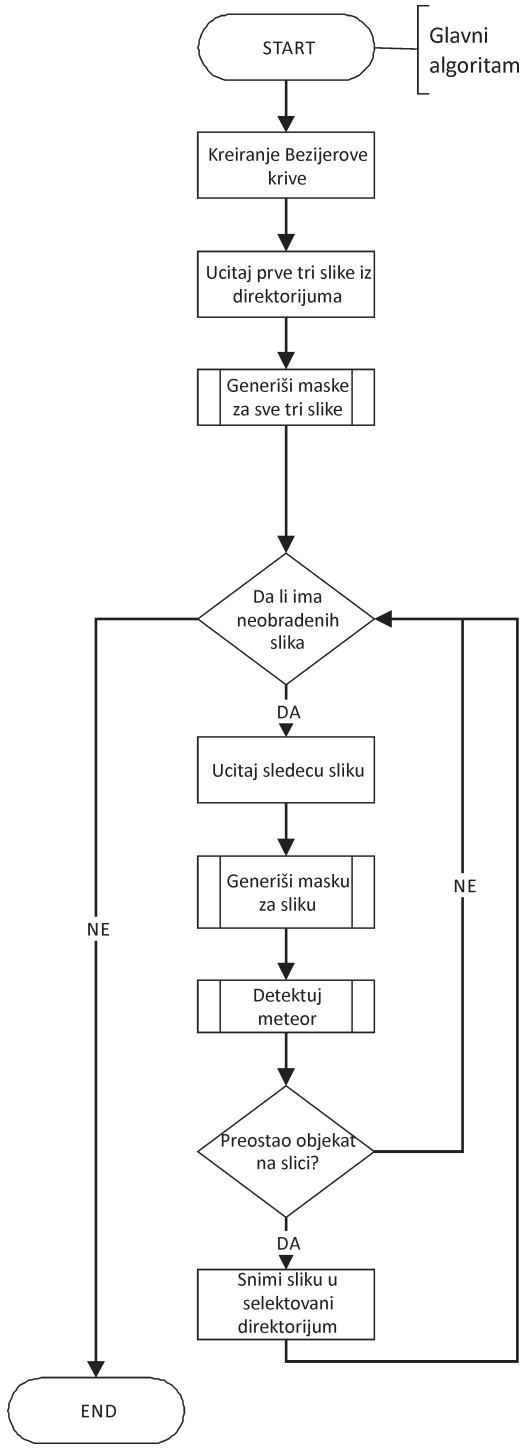
Figure 4. Enlarged section where a meteor is visible, present only on A-B and B-C (absent from C-D)

trom u centru slike. Sve izvan tog poluprečnika neće biti korišćeno u daljoj analizi. Zatim se vrši dilatacija koja svaki piksel proširuje za određeni broj piksela. Nakon što se generiše maska, obrađena slika se prebacuje u grayscale, posle čega se binarizuje ranije određenim trešholdom. Posle ove operacije, sa slike se brišu svi objekti čija je površina manja od neke konfigurabilne vrednosti. Na ovaj način se briše šum, mrtvi pikseli i



Slika 5 (gore). Algoritam za detekciju meteora

Figure 5 (above). Algorithm for meteor detection



Slika 6 (desno). Glavni algoritam

Figure 6 (right). Main algorithm

slično. Nakon što se ove operacije primene na sliku, slika je spremna za dalju analizu.

Algoritam za generisanje maske prikazan je na slici 1.

U datom primeru u seriji od četiri uzastopna snimka na slici 2, na snimku B (drugi po redu) nalazi se meteor. Kada se primeni prethodno opisani algoritam za čišćenje slike, dobiju se snimci koji sadrže samo zvezde. Nakon toga se snimci preklope sa susednim snimcima. Preklope se i njihove maske. Sve što je zajedničko masci i odgovarajućem snimku boji se u crno. Preklopljeni snimci izgledaju kao što je prikazano na slici 3. Ilustracija na slici 4 prikazuje uvećan deo preklopljenih snimaka koje sadrže meteor.

Na drugom snimku prikazan je meteor, a meteor se nalazi i na preklapanju prvog i drugog snimka i drugog i trećeg, ali ne i na preklapanju trećeg i četvrtog. Noćno nebo se rotira kao posledica Zemljine rotacije. To znači da se zvezde nikada neće potpuno poklapati na dva uzastopna snimka. Meteor se nalazi na istoj poziciji na dva snimka, a na trećem ga sigurno nema. U slučaju da neki objekat ispunjava ovaj uslov, može se pretpostaviti da je taj objekat meteor.

Shema na slici 5 prikazuje algoritam za detektovanje meteora. Na slici 6 prikazan je glavni algoritam koji opisuje kojim se redom koriste implementirane funkcije.

Implementacija aplikacije METeorDEteCTor (METDECT) je izvršena u programskom jeziku C#.

Klase

Osnovne klase koje su implementirane za ovu aplikaciju su Slika i Zvezda.

Zvezda je klasa koja sadrži definicije koordinata piksela preko kojih se ta zvezda prostire, njenu dužinu, njen centar, kao i parametre prave koja prolazi kroz nju i koja je najbliža njenim pikselima. Metode koje su implementirane su Zvezda i DeleteZvezda.

Zvezda je overloadovani konstruktor koji prima 0 parametara ili 4 parametra, a to su lista koordinata piksela preko kojih se zvezda prostire, njena dužina, njen centar i funkcija prave.

DeleteZvezda je metoda koja briše datu zvezdu iz memorije.

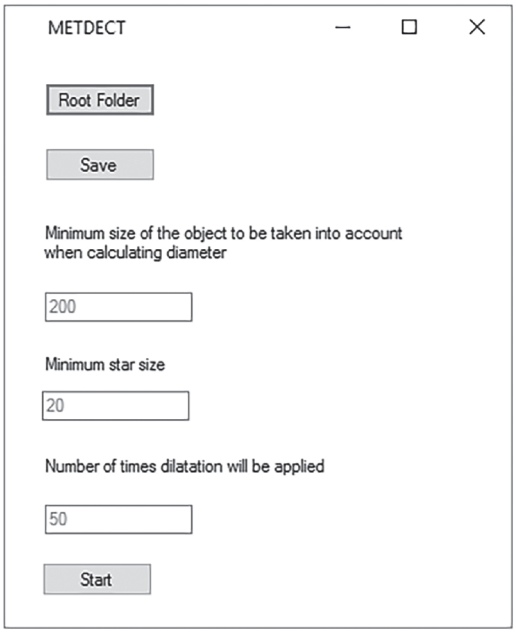
Druga klasa od značaja za objašnjenje implementacije je Slika. Ova klasa sadrži listu zvezda. Metode koje su implementirane za ovu klasu su AddZvezda i Check.

AddZvezda dodaje datu zvezdu u već postojeću listu zvezda.

Check je metoda koja proverava da li u blizini zvezde, u smeru u kojem ona ide, postoji još neka zvezda. Ako je to slučaj ta zvezda se ostavlja, inače se briše iz liste. U slučaju da je ostala bar jedna zvezda, originalna slika se snima u folderu koji je određen.

Korisnički interfejs

Korisnički interfejs aplikacije je jednostavan i sastoji se od tri dugmeta i tri tekst polja. Dugmad koja se nalaze na UI formularu su Open, Save i Start (slika 7). Kada se pritisne Open, otvara se folder browser dialog koji dozvoljava korisniku da izabere iz kog foldera će se slike čitati. Kada se pritisne Save, otvara se isti dijalog, i korisnik bira u koji folder želi da se slike snimaju. Tekst polja koja se nalaze na formularu služe za konfigurisanje veličine objekata koji će



Slika 7. Izgled korisničkog interfejsa aplikacije METDECT

Figure 7. Graphical User Interface for METDECT

se uzimati za računanje poluprečnika u masci, za određivanje minimalne veličine zvezde da se zvezda ne bi obrisala i koliko puta će se dilatacija izvršiti. U tekst poljima je već stavljen tekst koji reprezentuje podrazumevanu konfiguraciju.

Dobijeni rezultati

Prvobitna verzija aplikacije je izdvajala od 30 do 50 posto slika kao da sadrže pojavljivanje meteora. U ovoj verziji maska se generisala jednom i primenjivala isto na svaku sliku. Problem kod ovog pristupa je što prilikom velike oblačnosti ostaje puno šuma.

Finalna verzija aplikacije izdvaja od 0 do 5% slika kao slike koje sadrže mogući meteor, što predstavlja efikasnost programa 95–100%. Generisanje maske za svaku sliku je značajno smanjilo grešku. Oblik objekata koje program greškom izdvaja je ostao isti, na primer, problematični su objekti sa velikom površinom i rupama, a to mogu da budu oblaci, tragovi svetlosnog zagađenja i slično.

Testni primeri na kojima je program radio slikani su petničkom kamerom POgLED, njihova veličina je 5184×3456 piksela, a analizirano je 3390 slika od kojih samo jedna ima meteor. Testiranje je rađeno sa paralelnim radom tri nezavisne aplikacije, od kojih je svaka bila izvršavana na po jednom jezgru procesora i obrađivala po 1130 slika. Ukupno vreme obrade ovih slika je bilo 8.5 sati.

Računar na kojem je izvršavana aplikacija ima procesor Intel i5 na 3.1 GHz, 8 GB RAM, HDD 1 TB 7200 rpm.

Diskusija i budući rad

Jedan od načina na koji ovaj algoritam može da se poboljša je implementacija detekcije oblika. Većina slika koje aplikacija izdvaja greškom imaju objekte koji su, za razliku od objekata koji pripadaju meteoru, manje gusti. To znači da imaju više rupa u sebi i da su okruženi većim brojem crnih piksela, kao i da imaju veći obim za površinu koju pokrivaju. Generisanje maske za svaku sliku je znatno usporilo izvršavanje aplikacije, ali su se dobili kvalitetniji rezultati. Zbog dinamičkog generisanja maske, smanjen je broj tih objekata i nema ih puno – njihov broj na slici kreće se od 1 do 10. Ovo bi moglo da se reši tako što bi se povećao minimalni broj objekata koji čine meteor, ali bi to moglo da rezultuje u gubitku veoma brzih i kratkih meteora, zbog čega to u ovom projektu nije urađeno. Moguće je ubrzati rad aplikacije izvršavanjem obrade slika u memoriji grafičke kartice. Potrebno je nastaviti testiranje sa većim brojem slika, kako bi se dobili kvalitetniji rezultati analize rada aplikacije. Izvršavanje METDECT aplikacije bi moglo da se ubrza postavljanjem na cloud i izvršavanjem u formi nekoliko nezavisnih servisa.

Literatura

Bettonvil F. 2015. POgLED, Petnica bolide all-sky camera device. Instruction manual v1.0. Petnička meteorska grupa, Istraživačka stanica Petnica, 14000 Valjevo

Evich A. M. 2012. Automated detection and analysis of meteor events using nightly calibrations of observed stars. Physics student work, College of Saint Benedict and Saint John's University. http://digitalcommons.csbsju.edu/physics_students/1

García A. L. 2008. Detection of transient phenomena with fish eye cameras. *Astronomical Data Analysis Software and Systems ASP Conference Series*, **394**: 355.

Reffet B., Vaubaillon J., Colas F. 2014. A new meteor detection algorithm for shuttered photography. U *Proceedings of the International Meteor Conference, Giron, France, 18-21 September 2014* (ur. J-L. Rault i P. Roggemans). International Meteor Organization, str. 136-138.

Web 1. Wikipedia Grayscale https://en.wikipedia.org/wiki/Grayscale. Pristupljeno 23. 08. 2016.

Web 2. Jakub Szymanowski, Fundementals of Image Processing, http://www.codeproject.com/Articles/781213/Fundamentals-of-Image-Processing-behind-the-scenes. Pristupljeno 23. 08. 2016.

Milenko Gavrić

Meteor Detection Using a Series of Consecutive Images Taken by All-Sky Camera

In this paper the application METeorDEteCTor, which pins out pictures possibly containing meteors from the series of shots taken by all-sky camera with fish-eye lenses, was developed. Papers on the analytics and processing of pictures taken by cameras with fish-eye lenses were analyzed in great detail, as well as the therein applied algorithms, which were partly implemented, with adequate changes. A library in C# was developed for supporting the implementation of the defined algorithm. A class which contains general algorithms for processing images, implemented as methods, that were used in this project, was defined. The main algorithms that were used in this class are the algorithm for generating Béziers curve, algorithm for determining threshold, breadth first search, algorithm for finding a line that is closest to the given dots and dilatation. A separate class was implemented for storing the selected objects (possible stars and meteors), as well as their further testing. In addition, the Aforge.Imaging library was used for defining the needed palettes. The achieved results of an analysis of 3390 pictures taken by the Petnica all-sky camera are shown, along with suggestions for possible improvements.