Ognjen Tomcic 28/21 C

Prototip sistema za detekciju uslova otežane vidljivosti u procesu autonomne vožnje

Uvod:

Prije nego što počnem sa objašnjavanjem algoritma, neophodno je dati određene smernice i motivaciju da bi se algoritam u potpunosti razumio.

Funkcija cv.adaptiveThreshold() konvertuje sivoskaliranu sliku u binarnu, tako što vrijednost trenutnog piksela određuje na osnovu piksela koji ga okružuju. Ako se trenutni piksel ne razlikuje mnogo u intezitetu od okruženja, dodijeliće mu se ista binarna vrijednost kao susjednim pikselima.

Slike sa kamera motornih vozila uočavaju veliki broj objekata, kao što su druga vozila, stambeni objekti, drveća i mnogi drugi. Takvi objekti zbog mnogo detalja i ogromnih razlika u intezitetu boja između detalja, kad se primeni cv.adaptiveThreshold() postanu „išarani“ crno bijelim pikselima. Sa druge strane, ukoliko imamo mrlju na slici, njeni pikseli međusobno nemaju veliku razliku u intezitetu između sebe, što uzrokuje da oni budu „obojani“ isklučivo crnom ili bijelom bojom nakon cv.adaptiveThreshold() funkcije.

Jedno svojstvo binarnih slika jeste broj bijelih piksela. U ovom problemu, ovo svojstvo može da se iskoristi kako bi se slike razvrstavale na slike dobre i otežane vidljivosti. Kad sliku konvertujemo u binarnu koristeći cv.adaptiveThreshold(), ukoliko se na slici nalazila mrlja, to mjesto biće zamijenjeno pikselima crne boje, utičući da se broj piksela bijele boje smanji. Slike sa malim brojem bijelih piksela, ukazuju da postoji mrlja, zamagljenje ili bilo šta drugo što utiče na jasnoću kamere.

Ideja:

Osnovna ideja se sastoji da sliku, binarizujemo, potom da prebrojimo broj bijelih piksela i na osnovu njega zaključimo da li je vidljivost dobra ili otežana.

Kod:

Kod je napisan u dva Python fajla main.py i fun.py (funkcije). U fun.py su implementirane sve funkcije, a unutar main.py se poziva funkcija iz fun.py.

Fun.py ima funkciju pokreni(), koja ima parametar path, što je putanja do slike koju želimo da procijenimo. Ona poziva funkciju preprocesirajSliku() koja ima parametar path, čija je svrha identična kao kod prethodne funkcije. Ona pretvara sliku u sivoskaliranu sa cv.cvtColor() funkcijom, potom u binarnu pomoću cv.adaptiveThreshold(). Svaka konverzija slike se prikazje na ekranu. Kao return vrijednost vraća binarnu sliku.

Potom poziva daLiJeVidljivostDobra() koja uzima parametar image, što predstavlja binarnu sliku. Pomoću for petlji se prolazi kroz sliku i broje se bijeli pikseli. Ako je broj bijelih piksela veći od 111 000 (kasnije u sekciji Dodatno je objašnjeno zašto ovaj broj), smatra se da slika ima dobru vidljivost, inače nema. Respektivno ova funkcija vraća True, odnosno False.

Na kraju funkcija promijeni uzima return vrijednost prethodno pomenute funkcije i vraća originalnu sliku u boji sa odgovarajućim natpisom dobre ili loše vidljivosti.

Main.py je pxthon modul koji samo poziva funkciju pokreni() iz fun.py modula.

Evaluacija:

Moj sistem ima tačnost od 98% procenata. Bez greške predviđa slike dobre vidljivosti, dok samo jednu sliku nije u stanju da prepozna da je otežane vidljivosti. Razlog tome je zbog same prirode mog algoritma. On broji bijele piksele, pa ukoliko ima nekoliko mrlja, ali je slika velike jasnoće, postoji mogućnost da je protumači za sliku dobre vidljivosti. Kako bi dr ovaj propust uklonio ukoliko slika ima broj piksela blizu granice, odnosno 111000 piksela, da se uradi dodatna analiza.

Dodatno:

Ovaj sistem sadrži dodatne funkcije kao što je obradaFoldera(path1, path2). Ukoliko imamo folder sa slikama kamere, ova funkcija binarizuje sve slike iz datog foldera, koji je predstavljen parametrom path1, i premešta binarne slike u folder na lokaciji path2. Potom generiše tri fajla. U jednom fajlu za svaku sliku zapiše ime i broj bijelih piksela, drugi fajl je json fajl čija je struktra ista kao kod prvog fajla. Treći fajl ima osnovne obrađene podatke iz prethodna dva fajla kao što je maksimalan, minimalan i proskečan broj bijelih piksela kod obrađenih slika. Nazivi tako generisanih fajlova su “podaci\_o\_pikselima.txt”, “podaci\_o\_pikselima.json” i “statistika.txt”, redom. Na osnovu podataka iz “statistika.txt” uzet je neophodan broj piksela koji odeđuje da li slika ima dobru vidljivost,a to je minimalan broj piksela slike iz “dataset/good” foldera – 111 000 bijelih piksela.

Mane:

Mane mog sistema su sporost tokom procijene slike i potencionalan overfitting sistema na dataset koji je dat za evaluaciju.

Algoritam je spor zato što broji bijele piksele, piksel po piksel. Ovaj pristup rezultira u 500 000 iteracija nad np.array(), što crpi resurse računara. Da se popravi ovaj propust moguće je umesto da se ide piksel po piksel, uzima matrica nxn dimenzija, gdje je n mali prirodan broj i nasumično bira piksel iz matrice, potom da se pretpostavi da je data matrica u potpunosti ispunjena bojom piksela koji je nasumično izabran. Na ovaj način bi se povećala brzina izvršavanja, ali preciznost bi se smanjila.

Drugi problem je overfitting na dataset. Slike iz njega imaju neke zajedničke karakteristike koje nemaju nužno ostale slike. Na primjer, donji dio bilo koje slike sadrži crne piksele koji ne utiču na procijenu vidljivosti, vjerovatno je u pitanju objektiv kamere. Takođe gornji dio slike je zanemaren, odnosno nebo, jer ukoliko je ovaj dio zamagnjen, on zapravo ne utiče na vidljivost, pošto je drum i njegova okolina vidljiva. Iz ovih razloga gornji i donji dio slike je zanemaren, što ne mora biti slučaj sa ostalim kamerama. Druga karakteristika koja utiče na overfitting jeste rezolucija koja je ista kod svake slike. Kod lika drugih kamera granični broj 111000 može biti pogrešan.