Ugrađeni sustav za praćenje prometa i kontrolu prometnih nezgoda

Starčević, Matija

ISTARSKO VELEUČILIŠTE –

UNIVERSITÀ ISTRIANA DI SCIENZE APPLICATE

Matija Starčević

UGRAĐENI SUSTAV ZA PRAĆENJE PROMETA I KONTROLU PROMETNIH NEZGODA

Završni rad

Pula, 2022

ISTARSKO VELEUČILIŠTE –

UNIVERSITÀ ISTRIANA DI SCIENZE APPLICATE

Matija Starčević

UGRAĐENI SUSTAV ZA PRAĆENJE PROMETA I KONTROLU PROMETNIH NEZGODA

Završni rad

JMBAG: 0233008077, redoviti student

Studijski smjer: Preddiplomski stručni studij Mehatronika

Predmet: Projektiranje ugrađenih računalnih sustava

Mentor: Marko Turk, dipl. oec., pred.

Pula, 2022

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Matija Starčević, kandidat za prvostupnika Mehatronike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitoga rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuje korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

U Puli, 2022. godine

Student

_____

IZJAVA

o korištenju autorskog djela

Ja, Matija Starčević dajem odobrenje Istarskom veleučilištu – Universita Istriana di scienze applicate, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom Ugrađeni sustav za praćenje prometa i kontrolu prometnih nezgoda koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____

Potpis

_____

SAŽETAK

Cilj ovog rada je pokazati na jednostavan način detekciju i praćenje prometa uz pomoć Python programskog jezika i njegovih biblioteka te OpenCV biblioteke. U radu imamo dvije podjele, teorijski dio i praktični dio. U teorijskom dijelu rada opisano je trenutno stanje inteligentnih transportnih sustava te njihova primjena, uređaji, algoritmi i metode koji se koriste u detekciji i praćenju programa. U praktičnom dijelu rada odrađeno je rješavanje problema detekcije i praćenja. Detaljno je objašnjen kod te logika iza njega. Detekciju smo uspostavili sa bibliotekom OpenCV Python programskog jezika, a praćenje se odradilo sa klasifikacijom vozila i logikom koja je povezala sve u jednu cjelinu.

KLJUČNE RIJEČI

Python, OpenCV, detekcija, praćenje, promet

ABSTRACT

The aim of this paper is to easily detect and monitor traffic through Python programing language and its libraries and OpenCV Library. In this paper we have two divisions, a theoretical part and a practical part. The theoretical part of the paper describes the current state of intelligent transport systems and their applications, systems, algorithms and methods used in programming detection and monitoring of traffic. The practical part of the paper deals with the problem of detection and monitoring and is explained in detail with the logic behind it. The detection was established with the OpenCV library and Python programming language, and the monitoring was done with vehicle classification and logic that connected everything together.

KEYWORDS

Python, OpenCV, detection, tracking, traffic

Sadržaj

1. Uvod

Ugrađeni sustavi za praćenje prometa su sustavi kojima se iz dana u dan pridaje sve veća pažnja. Sa stalnim porastom broja vozila u prometu dolazimo do potrebe za implementacijom Inteligentnih transportnih sistema (ITS) za rješavanje problema poput gustoće prometa, brzina prometa, duljina čekanja, ukupnog broja vozila koja prolaze kroz točku u određenom vremenu itd.. Inteligentnim transportnim sistemom snimanja cestovnog prometa putem kamera možemo uvelike olakšati praćenje prometa i njegovu kontrolu.

U ovom se radu valja usredotočiti na dio ITS-a koji se bavi praćenjem cestovnog prometa pomoću kamera postavljenih na autoceste te prikupljanje podataka i obrada istih. Cilj rada je prikazati praćenje prometa pomoću dostupnih video materijala, prikupiti podatke sa video materijala i pomoću biblioteke OpenCV i Python programskog jezika napraviti sustav koji će detektirati i pratiti promet.

Pri izradi završnog rada koristila se dostupna literatura, prvenstveno istraživački članci zatim provjereni internet izvori. Također, važno je napomenuti kako je većina dostupne istraživačke literature bila na engleskom jeziku.

Važno je napomenuti kako se rad sastoji od sedam poglavlja. Osim uvodnog poglavlja, drugo poglavlje daje nam uvid cestovne transportne sustave te pregled nekoliko sustava koji su vezani za praktični dio rada zajedno s poglavljem koje opširnije prikazuje CCTV sustav i njegove glavne komponente. Četvrto poglavlje prikazuje nam vrste detekcije prometa i uvodi nas u osnovne komponente strojnog učenja. Peto poglavlje daje nam uvid u alate koje smo koristili te programski kod koji je napisan za ovaj sustav. U šestom poglavlju prikazujemo pogreške sustava te kako bi se one mogle ispraviti. Rad završava s poglavljem zaključka.

2. Inteligentni Transportni Sustavi u cestovnom prometu

Zbog velike važnosti koju dajemo cestovnom prometu i zbog sve veće količine motornih vozila u prometu transportni cestovni sistemi su od velike važnosti u modernom životu. Zbog toga sve razvijenije zemlje svijeta ulažu velike količine resursa u tehnologije transporta i kontrole prometa.

ITS[1] nastoji spojiti nove informacijske tehnologije za simulaciju, komunikacijske mreže te kontrolu u stvarnom vremenu kako bi izgradili infrastrukturu u kojoj bi se efikasno smanjilo zagađenje zraka, potrošnja goriva, smanjile prometne gužve i ono najvažnije, poboljšala sigurnost u prometu. Cestovni dio ITS-a sastoji se od više manjih složenih sustava te svaki od tih sustava ima svoju svrhu. Svaki sustav zasebno ma koliko složen, u ITS-u, nema toliku moć dok se ne poveže u jednu cjelinu.

Kako bi cestovni ITS sustav mogao pravilno funkcionirati, podijeljen je u tri osnovne funkcije rada, a to su: prikupljanje podataka, obrada podataka te prosljeđivanje podataka. Informacije koje cestovni sustav prikuplja mogu biti: brzina vozila na označenim dijelovima ceste, registarske oznake vozila, prepoznavanja nedozvoljenog korištenja mobilnih uređaja za vrijeme upravljanja vozilom, prepoznavanje mogućih zastoja i incidenata. Svi ti podaci se zatim šalju u centar za obradu podataka, gdje ih zatim obrađuju računala i uspoređuju sa zadanim parametrima. U slučaju da je dobivena informacija prekršila zadane parametre, ona se dodatno obrađuje i osoba koja je napravila prekršaj snosi sankcije. U Hrvatskoj primjer prosljeđivanja podataka vrši se tako da osobi koja je napravila prekršaj, na kućnu adresu šalje se kazna, dok u razvijenijim zemljama pomoću sustava za odašiljanje, možemo upozoriti osobu direktno u vozilu.

ITS sustav zapravo obuhvaća sve vrste prometa (cestovni, zračni, pomorski) i ono što ga čini moćnim je međusobna komunikacija. U ovom radu fokus je usmjeren na cestovni promet, prikupljanje podataka i njegovo praćenje.

[1] ITS – eng. Inteligent transport systems, Inteligentni trasportni sistemi Faletar M, Osijek, "INTELIGENTNI TRANSPORTNI SUSTAVI", 2020 , str 3-5

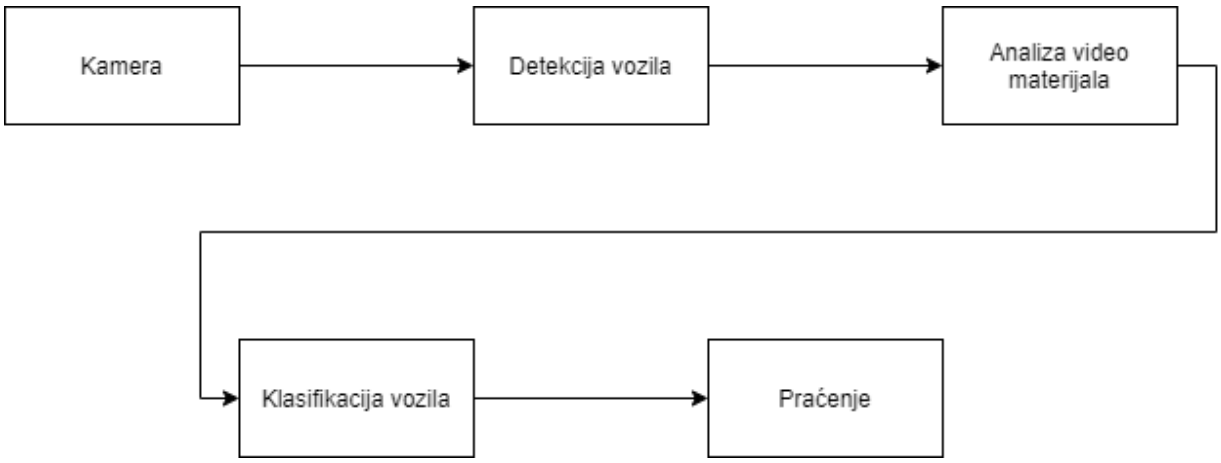
2.1. Tehnologije Inteligentnih transportnih sustava

Postoji cijeli raspon tehnologija koji se koriste u prometu i njegovoj kontroli. Neke od tih tehnologija su: automatsko prepoznavanje registarskih oznaka, kamere za praćenje brzine, detekcija prometnih nezgoda, automatsko preporučene brzine vozila itd. Svaka od tih tehnologija posjeduje sustav povratne informacije u kojemu se mjerenjem trenutnih podataka i povratnih veza između više sustava omogućuje napredno modeliranje i vođenje takvog sustava. U nastavku slijedi nekoliko tehnologija koje se trenutno koriste u prometu.

2.1.1 Kamera – trenutna video detekcija

Jedan od oblika praćenja prometa je automatska detekcija prometa kroz video materijal koji se odašilje u trenutnom vremenu u centralu [4]. Ovakva vrsta sustava se smatra nenametljivom vrstom sustava jer se ne postavlja na kolniku, već se postavlja na rasvjetne stupove ili stupove koji drže semafore.

Ovaj sustav sastoji se od dva dijela. Prvi dio je video materijal koji kamere prikupljaju na svojim lokacijama te ga šalju u centar na obradu. Drugi dio sustava sastoji se od računala koji imaju u sebi imaju procesor slike kojemu se zadaju početni parametri kao npr. razmak između linija, visina od kolnika do određene točke mjerenja itd. Takav sustav, ovisno o kompleksnosti video procesora, može prepoznati i klasificirati vozila, brojati vozila i zapisivati tablice.



Slika 1. Blok dijagram otkrivanja i kategorizacije vozila

2.1.2 Prepoznavanje registarskih oznaka na vozilu

[2]Još jedna metoda usko povezana uz [3.1.1] je metoda prepoznavanja registarskih oznaka na vozilima. Zbog svoje jednostavnosti primjena ovog sustava prilično je raširena u prometu i najčešće je viđamo na auto putevima ili na gradskim parkinzima. Kako bi sustav mogao funkcionirati potrebno mu je zadati ograničenja prema kojima će taj sustav funkcionirati. U slučaju da ograničenja nisu zadovoljena, djelotvornost sustava se smanjuje. Kako bi dobili rezultate sustav prolazi kroz nekoliko procesa obrade video materijala. Prvi dio je pred-obrada (Engl. Pre-processing). Ona se izvodi tako da se video materijal u boji pretvara u video sivih tonova. Razlog tomu je taj što eliminiranjem boja lakše možemo detektirati rubove i postaviti granice za detekciju vozila te samim tim registarskih oznaka. Nakon obrade kreće detekcija registarskih oznaka i ona se izvodi pomoću raznih analiza video materijala koje su specifično dizajnirane za pronalazak oznaka kao npr. morfološka operacije slike u kojoj se detektiraju objekti u slici (slova i brojevi) te siva u binarnu operacija u kojoj iz slike dobivamo vrijednosti 0 za crnu boju, 1-255 za bijelu boju. U nastavku slijedi par fotografija koje prate taj proces.

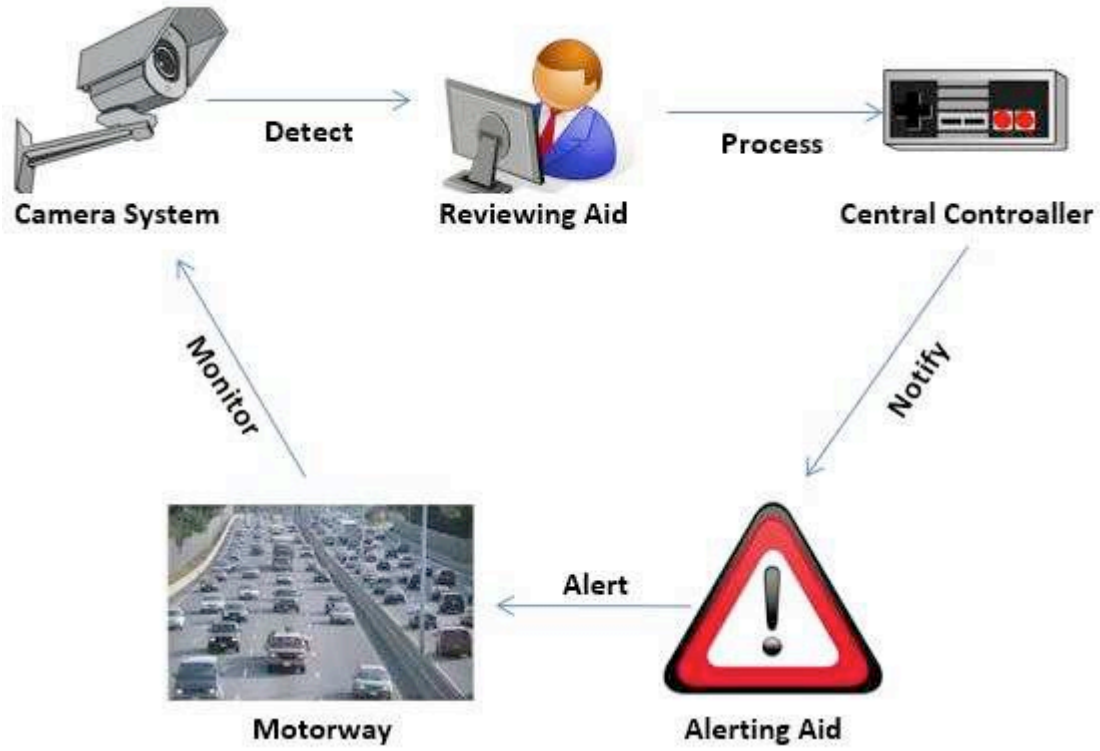
Slika 2. Prijelaz iz slike u boji u sliku sa sivim tonovima (Pavani, Mohan, 2019)

Slika 3. Ulazna slika, morfološka operacija slike, siva u binarnu sliku(Pavani, Mohan 2019)

[2] Mufti N, Shah S, 2021, „Automatic Number Plate Recognition: A Detailed Survey of Relevant Algorihms", str 1-35

3. CCTV sustav

CCTV (engl. Closed circuit television camera) televizijska kamera zatvorenog kruga je vrsta video kamere korištena u sustavima za praćenje prometa. Video kamere se smatraju modernim rješenjem nadzorne tehnologije. Kao što smo već spomenuli ovakve kamere postavljaju se na vrhove semafora ili rasvjetnih stupova gdje želimo nadzirati promet. U početcima, glavni fokus video praćenja prometa bio je na prikupljanju registarskih oznaka na vozilima, dok danas napretkom tehnologije, a ponajviše razvojem sustava za obradu video materijala, možemo prikupljati mnogo više informacija kao npr: oblik, veličina, boja, stanje okoline itd.



Slika 4. [3]Glavne komponente i tijek rada u CCTV sustavu za kontrolu prometa(Kurdi H, 2014)

[3] Kurdi H, April 2014. "Review of Closed Circuit Television (CCTV) Techniques for Vehicles Traffic Management", International Journal of Computer Science & Information Technology, str 3, slika 2.

Glavni dijelovi CCTV sustava kao što možemo vidjeti na slici 4 su : (a) Sistem kamere, (b) Pomoć pri pregledavanju, (c) Centrala, (d) Dodatci

(a) – Kako bi dodatno pojasnili sustav, kamere koje se koriste mogu se podijeliti na dvije varijante: automatsko snimanje i neautomatsko snimanje. Razlika među ove dvije metode je u automatskom snimanju kod kojega u slučaju da se ispred kamere ništa ne dešava ona prestaje snimati i time štedi energiju i skladištenje.

(b) – Pod pomoć pri pregledavanju zapravo spadaju programi koji spremaju snimljene video materijale u realnom vremenu kako bi, ako je potrebno, kasnije mogli radnici pregledati video materijal.

(c) – Centrala je zadužena za automatsku obradu video materijala i praćenje prometa, te po potrebi kontrole prometa.

(d) – Pod dodatke spadaju, kao što smo već spomenuli u (a), senzori za detekciju pokreta. Tu također spada i noćni vid koji se također može dodatno instalirati ukoliko je potrebno, jer sve novije kamere dolaze za već instaliranim noćnim vidom.

4. Detekcija i praćenje prometa

Nakon što smo uspješno postavili video kamere na pozicije slijedi detekcija vozila pomoću prikupljenog video materijala. Kako bi uopće procesor za obradu video materijala mogao prepoznati vozila mi mu zadajemo početne parametre kako bi ga „naučili“ što treba gledati i na što treba obratiti pažnju.

Sam sustav je vrlo kompleksan zbog „smetnji“ koje se dešavaju u okolini, a to znači promjena svjetline, sjena, promjena vremena itd. Kako bi lakše razumjeli cijeli sustav u ovom poglavlju krećemo s dijelom AI učenja (engl. Artificial intelligence learning).

Strojno učenje opisuje prepletanje računalne znanosti i statistike gdje se algoritmi koriste za izvršavanje zadataka bez izričitog programiranja, te tako prepoznaju obrasce u podacima i rade predviđanja čim novi podaci stignu. Postoje tri kategorije strojnog učenja: nadzirano učenje, učenje bez nadzora te pojačano učenje (engl. Supervised learning, unsupervised learning, reinforced learning).

4.1. Nadzirano učenje

Nadzirano učenje koristi već poznati odnos između ulaza i izlaza. To znači da algoritam „uči“ iz podataka koje smo mu mi zadali te tako radi predviđanja kada mu se pruže novi podaci. Postoje dva glavna zadatka nadziranog učenja:

a. Regresija – u većini slučajeva koristi se za razumijevanje odnosa između zavisnih i nezavisnih varijabli. To znači da mijenjanjem nezavisne varijable utječemo na zavisnu varijablu, te se tako one mogu pratiti kao uzrok i učinak.

b. Klasifikacija – je zapravo dodjeljivanje oznaka. Koristi se algoritam koji iz zadanih ulaznih i izlaznih vrijednosti pokušava izvući zaključke kako nešto definirati ili klasificirati.

Kako bi pokazali primjer za nadzirano učenje povezati ćemo ga sa trenutnom temom i to sa klasifikacijom slika. Možemo zamisliti da imamo set slika među kojima su slike

automobila i kamiona. Umjesto da sve slike označavamo sami izgradit ćemo algoritam koji to radi za nas. [4]

Prvo kreiramo mape i označimo ih sa nazivom točno te ih dodijelimo na ulaz i izlaz modela. U mape postavimo slike sa automobilima i kamionima i označimo ih s odgovarajućim nazivima (automobil ili kamion).

Drugo, u model dodajemo druge slike koje nisu označene te algoritam počinje „vidjeti" razlike između ulaza i izlaza koje smo mi označili kao točne. To može biti razlika u veličini, razlika u razmaku između guma itd.

Treće testiramo model na podacima koje smo mu dali i gledamo koliko je točnost predviđenog.

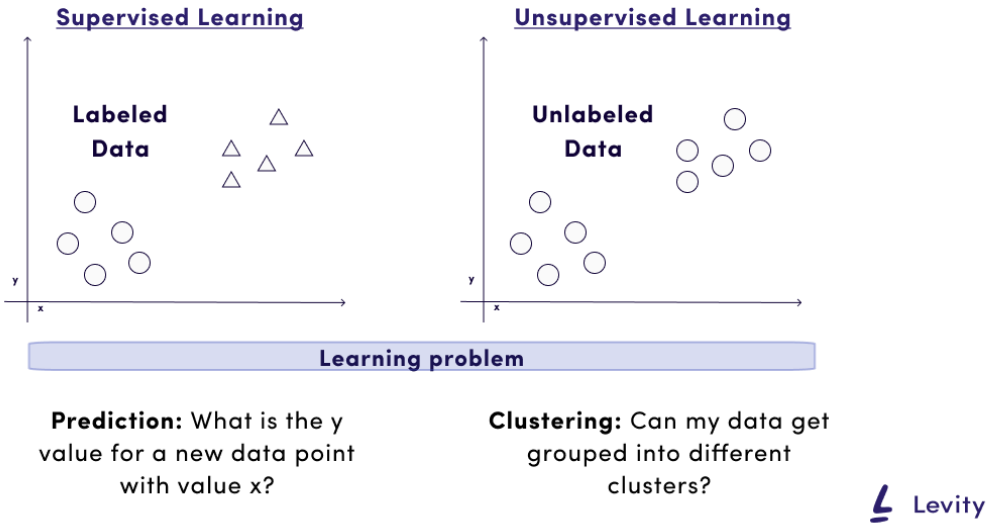
Ključna razlika između nadziranog i nenadziranog učenja je ta što u nadziranom učenju mi zadajemo skup podataka koje označavamo kao točne kako bi algoritam mogao dalje sam određivati – klasificirati podatke.

[4] Fritz Labs, „Image Recognition guide", [https://www.fritz.ai/image-recognition]

4.2. Ne nadzirano učenje

Ne nadzirano učenje se razlikuje od nadziranog u tome što u ne nadziranom učenju zadajemo samo ulazne podatke koji nisu označeni. Ovakva metoda prikladna je kada ne znamo kako bi naši rezultati trebali izgledati.

Cilj ovog algoritma je steći znanje i napraviti strukturu (grupirati podatke) sa zadanim podacima koji nisu označeni.. U nastavku je primjer koji prikazuje razliku između nadziranog i ne nadziranog učenja.



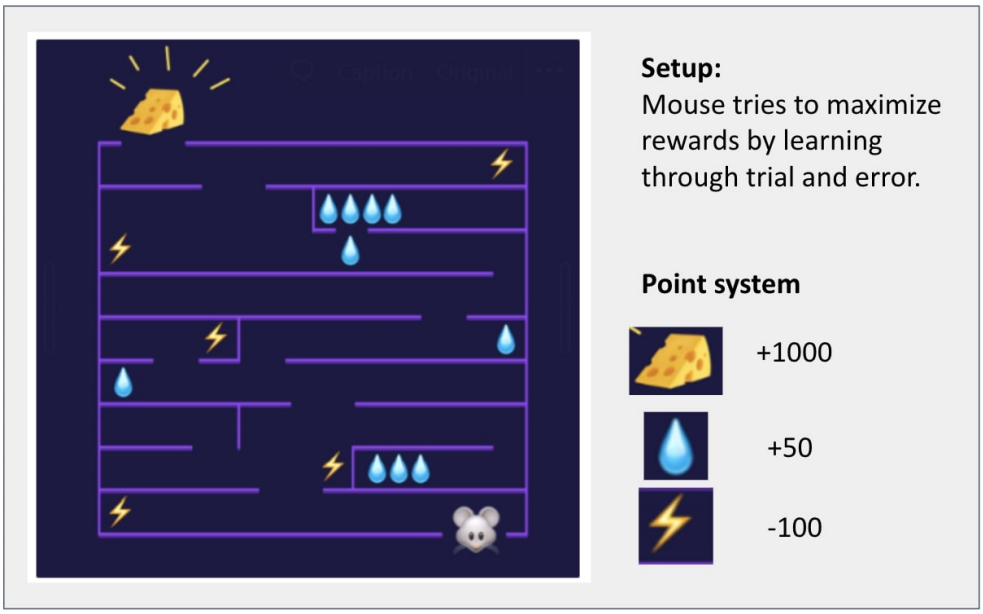
Slika 5. Prikaz razlike između nadziranog i ne nadziranog učenja (Wolfewicz A, 2021)[5]

[5] Wolfewicz A, (2021), "A begginer's guide to how machines learn" Raspoloživo na [https://levity.ai/blog/how-do-machines-learn]

4.3. Pojačano učenje

[6]U pojačanom učenju algoritam uči na načinu pokušaja i pogreške i koristi se povratnom vezom (informacijom) kako bi poboljšao svoj rad. U ovom modelu postavljaju se nagrade i kazne u vidu bodova kako bi algoritam „znao“ koje je željeno ponašanje a koje nije.

U nastavku slijedi primjer pojačanog učenja na jednostavnom primjeru miša kroz labirint.



Slika 6. Primjer pojačanog učenja na primjeru labirinta (Wolfewicz A, 2021)

Kao što vidimo iz primjera, miš pokušava doći do kraja labirinta i postići maksimalan broj bodova. Kako bi to postigao potrebno je izbjeći munje, a skupiti sve kapljice te doći do sira. Algoritam prilikom skupljanja munje dobije negativnu povratnu vezu i od nje uči te tako se svakim pokušajem poboljšava.

Razlika pojačanog učenja od nadziranog učenja je u tome da kod nadziranog učenja koristimo metodu u kojoj mi zadajemo točne odgovore te algoritam može usporediti

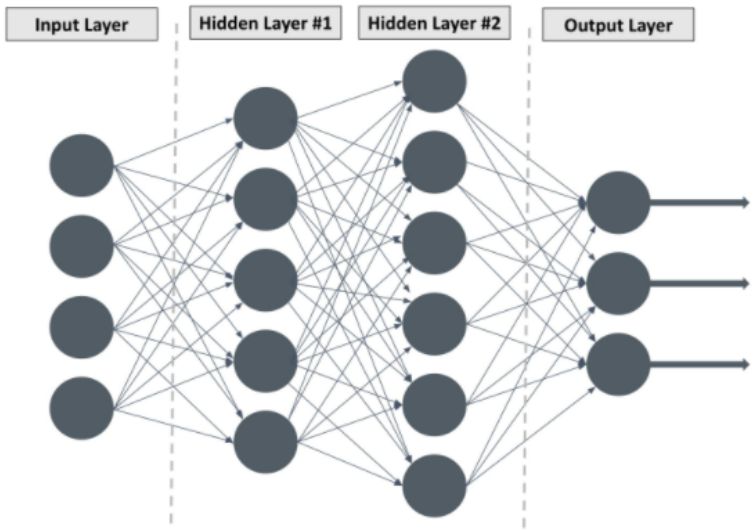
[6] Wolfewicz A, (2021), "Deep learning vs. machine learning – What's the difference?" Raspoloživo na [https://levity.ai/blog/difference-machine-learning-deep-learning]

točne odgovore sa ulazima te tako doći do zaključka, dok pojačano učenje koristi metodu povratne veze (pokušaj/pogreška) kako bi došli do željenih rezultata. Razlike pojačanog učenja od ne nadziranog učenja je u ciljevima. Cilj ne nadziranog učenja je grupirati podatke te ponuditi neki izlaz ovisno o ponuđenim ulazima, dok kod pojačanog učenja cilj je maksimizirati nagradu tj. prikupiti što veći broj bodova.

4.4. Duboko učenje

[7]Duboko učenje (engl. Deep learning) je dio umjetne inteligencije koji se smatra sofisticiranom i matematički kompleksnom evolucijom strojnog učenja. Slično kao što čovjek logički zaključuje tako i algoritam analizira podatke i donosi logične zaključke. Takvo učenje može se dešavati kroz nadzirano ili ne nadzirano učenje. Kako bi postigli ovakvu kompleksnost učenja duboko učenje koristi strukturu algoritama zvanu umjetna neuronska mreža (engl. Artificial neural network).

Umjetna neuronska mreža inspirirana je biološkom neurnonskom vezom ljudskog mozga, što ga stavlja u poziciju učenja daleko više od strojnog učenja. Kako bi detaljnije mogli objasniti kako izgleda ta veza u nastavku slijedi fotografija umjetne neuronske mreže.



Slika 7. Prikaz pojednostavljene umjetne neuronske mreže(Wolfewicz A, 2021)

[7] Wolfewicz A, (2021), "Deep learning vs. machine learning – What's the difference?" Raspoloživo na [https://levity.ai/blog/difference-machine-learning-deep-learning]

Kao što možemo vidjeti na slici lijeva strana slike prikazuje nam ulaze u sustav a desna strana prikazuje nam izlaze iz sustava. Ulazi i izlazi iz sustava mogu biti različiti ovisno o tome što želimo postići. Srednji sloj naziva se i skriveni sloj zbog toga što prilikom rada na algoritmu ne možemo pratiti vrijednosti u njemu. Što je više takvih skrivenih slojeva algoritam je kompleksniji i bolji. Svaka umjetna neuronska mreža koja ima više od dva skrivena sloja smatra se dubokom neuronskom mrežom.

U nastavku, spomenuti ćemo neke od jednostavnijih metoda kojima se možemo koristiti u detekciji i praćenju prometa.

4.5. Metoda sjena

Možemo reći da je metoda sjena jedna od jednostavnijih u nizu metoda koje se koriste u detekciji vozila u prometu. Svako vozilo u prometu ostavlja određenu veličinu sjene i ona se može softverski prepoznati i označiti. Kako bi dobili prepoznavanje pomoću sjena video materijal se iz videa u boji prebacuje u video sivih nijansi. Razlog tomu je taj da sve što se u sceni ne miče možemo obojati u crnu, a objekti koji se pomiču možemo obojati u bijele nijanse. Nakon te pretvorbe u programu si zadamo koliko piksela će se prikazivati oko objekta i dobijemo jednostavan oblik praćenja vozila.

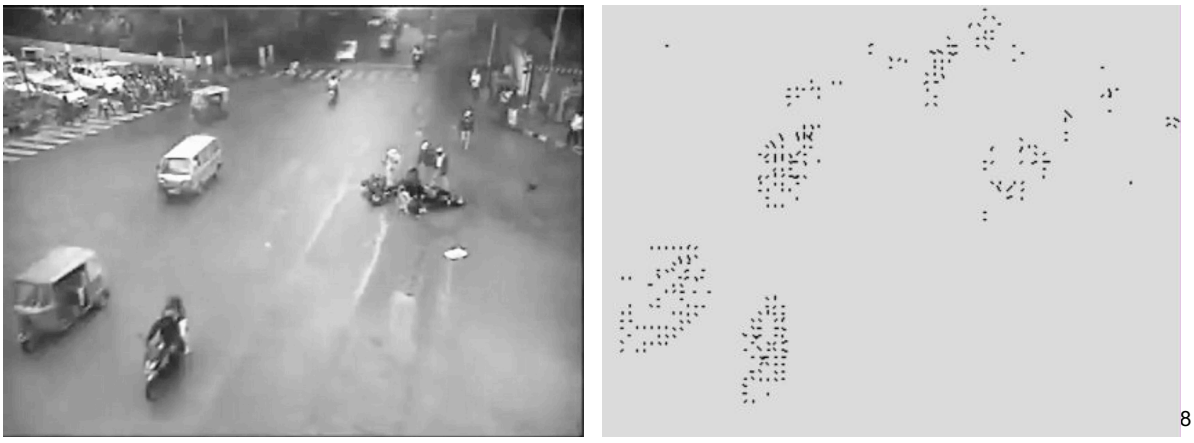
Slika 8. Prikaz metoda sjena

4.6. Metode temeljene na kretanju – optički tok

Ova metoda praćenja prometa jedna je u nizu metoda koje se koriste u kombinaciji s drugim metodama i temelji se na proračunu optičkog protoka. Ona je jedna od važnijih i složenijih metoda u ITS-u. Optički tok predstavlja očitu promjenu pomičnog objekta između slika u videu. Cilj optičkog toka u detektiranju vozila je razdvajanje pokretnih vozila od statične pozadine.

Kako bi ova metoda bila učinkovita prije implementiranja algoritma potrebno je predobraditi video materijal. Prvo se video postavlja u nijanse sive kao što je već spomenuto u [4.1.1] te se nakon toga primjenjuje medijanski filter (engl. Median filter) koji ima funkciju uklanjanja smetnji u video materijalu.

Optički tok opisuje smjer i brzinu piksela u vremenskom slijedu od dvije slijedne slike. Dvodimenzionalni vektor brzine koji nosi informacije o smjeru i brzini kretanja dodijeljen je svakom pikselu na određenom mjestu slike. Kako ova vrsta detekcije zahtjeva puno računalne snage, 3D objekti se prenose u 2D objekt. Tada sliku možemo opisati pomoću 2D dinamičke funkcije svjetline, mjesta i vremena. U nastavku slijedi primjer ovakve obrade video materijala.



(a) (b)

Slika 9. (a) Primjer video materijala u sivim tonovima, (b) Prikaz vektora gibanja nakon obrade optičkog toka (Aslani, Mahdavi, 2013)

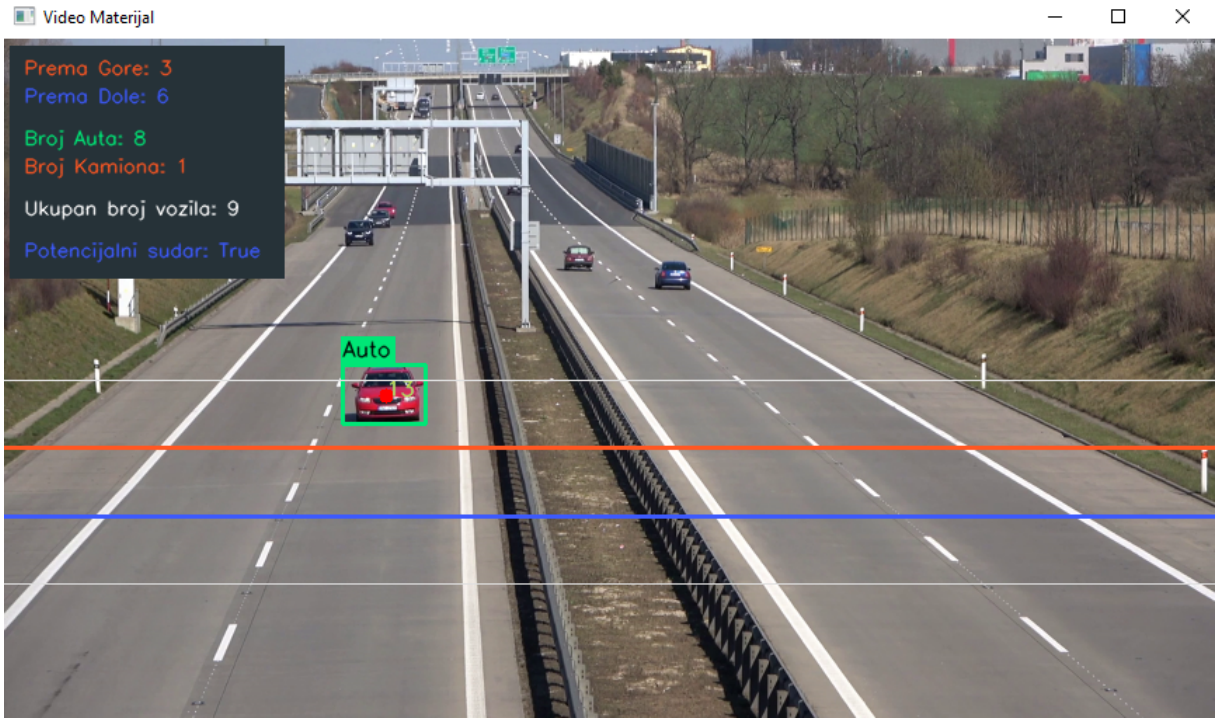
[8] Aslani S, Mahdavi-Nasab H, 2013, "Optical Flow Based Moving Object Detection and Tracking for Traffic Surveillance" – str 4, slika 3.

4.7. Kontrola prometnih nezgoda

Kako bi upotpunili ovaj sustav potrebno mu je ugraditi sustav kontrole prometnih nezgoda. Danas postoji veliki broj sustava za kontrolu i sprječavanje prometnih nezgoda koje koristimo svakodnevno i ne obraćamo pozornost na njih. Neki od tih sustava su : sigurnosni pojas, zračni jastuci, ugrađeni sustavi za izbjegavanje sudara u automobilima, sustavi za upozoravanje od potencijalnog sudara itd.

U ovom radu bavit ćemo se sustavom prepoznavanja mogućnosti prometne nezgode iz video materijala kojeg dobivamo uživo sa mjesta gledanja i ranog javljanja centrali kako bi pregledali video materijal i odlučili je li potrebno reagirati na uzbunu koju je sustav dao ili ne.

U sustav je implementiran kod koji detektira vozila koja su zaustavljena na dijelovima autoceste na kojima je zabranjeno zaustavljanje, te tako prepoznaje i daje signal za provjeru tog video materijala.



Slika 10 Prikaz detekcije potencijalnog sudara

5. Detekcija prometa pomoću OpenCV biblioteke i programskog jezika Python

Kako bi krenuli na praktični dio rada prvo se moramo ukratko upoznati s alatima s kojim smo odradili završni dio rada. Koristili smo Python programski jezik te OpenCV biblioteku koju smo uvezli u Python.

OpenCV[9] (engl. Open computer vision) je biblioteka otvorenog koda, a to znači da je bilo tko može preuzeti i raditi s njom ili je nadograđivati. Fokus OpenCV biblioteke je rad s podacima u stvarnom vremenu (real-time aplikacije). Cilj ove biblioteke je ostvarivanje jednostavnijeg i bržeg razvoja aplikacija te pružanje jednostavnije infrastrukture za rad s računalnim vidom (eng. Computer vision).

U ovom radu OpenCV biblioteka koristila se u svrhu detektiranja i brojanja vozila u prometu.

Python je višenamjenski programski jezik koji se može primijeniti u bilo kojem dijelu gdje se koriste podaci, matematika ili kod. To znači da python nije ograničen na jednu djelatnost npr. razvijanje web stranica ili razvoj aplikacija. Python radi u paru s tumačem koji izvršava kod red po red što ga čini jako privlačnim kod mnogih programera zbog jednostavnosti ispravke koda. Može se koristiti za najjednostavnije zadatake kao npr. programiranje botova (robota koji služe za automatkse odgovore ili razgovore), do vrlo kompleksnih kodova za analiziranje financijskih podataka. Postoji veliki broj resursa za učenje python-a i spada pod najjednostavniji jezik za učenje i čitanje. Zbog svih prednosti od prije u tekstu i zbog jednostavnosti implementacije OpenCV biblioteke u Python je odabran on kao programski jezik.

Važno je napomenuti kako se ovakva vrsta koda najčešće koristi u stvarnom vremenu iz prijenosa uživo.

[9] OpenCV – Otvoreni računalni vid

5.1. Uvod u praktični dio

Kako bi lakše pratili dalji razvoj praktičnog dijela ovaj dio biti će kratki pregled koraka koje smo donosili u razvoju praktičnog dijela. Praktični dio dijeli se na tri dijela tj. dijelove programa.

U prvom dijelu koda pod nazivom „my_ekran.py“ vrši se pripremni dio koda u kojem smo definirali linije detekcije koje smo koristili za razvrstavanje i brojanje prometa, tekst koji je prikazan u gornjem lijevom uglu na video materijalu te pravokutnik i krug koji nam služe kako bi vizualno detektirali i obilježili promet.

Drugi dio koda pod nazivom „my_vozilo.py“ je također dio pripremnog koda u kojem smo klasificirali vozila i postavili uvjete po kojima ćemo pratiti prolazak prometa.

Treći dio koda naziva „main.py“ sastoji se od uvoza svih preostalih potrebnih biblioteka, definiranja svih ulaznih varijabli te petlje koja se izvršava neprestano dok traje video.

U nastavku detaljnije ćemo pogledati svaki dio programa posebno i pojasniti sam kod te procese koji se događaju.

5.2. „my_ekran.py“ Kod

Kako bi mogli krenuti sa bilo kakvom radnjom, potrebno je uvesti biblioteke koje ćemo koristiti u tom dijelu programa. U my_ekran fajlu koristili smo dvije biblioteke naziva OpenCV te biblioteku NumPy.

```
import cv2
import numpy as np
```

Biblioteke uvozimo sa naredbom „import“ te pišemo naziv koji se koristi za uvoz. U slučaju OpenCV biblioteke pišemo cv2, a u slučaju NumPy biblioteke pišemo numpy. Kako bi si skratili pisanje koda u nastavku možemo dodati naredbu „as“ sa kojom možemo skratiti riječ u našem primjeru numpy --> np .

Kao što smo već spomenuli u [5], OpenCV nam je biblioteka koja nam daje funkcije kako bi mogli raditi sa podacima u stvarnom vremenu. Sa druge strane NumPy biblioteka daje nam podršku za višedimenzionalne nizove i matrice s velikom kolekcijom matematičkim funkcija/operacija za rad na tim matricama.

U nastavku koda definiramo ulazne varijable :

```
font = cv2.FONT_ITALIC
bg_rect_color = (56, 50, 38)
```

Zadajemo font koji ćemo koristiti u prikazu rezultata te boju pozadine na kojem će se prikazivati rezultati.

```
line_down_color = (254, 90, 61)
line_up_color = (34, 87, 255)
auto_boja = (118, 230, 0)
kamion_boja = (34, 87, 255)
accident_warning_color = (254, 90, 61)
```

Ovim linijama koda zadajemo boje za linije koje se prikazuju na videu, te boje okvira oko detektiranih vozila.

```
def getDetectionLinePointsMatrix(lineY, width):
    point_1 = [0, lineY]
    point_2 = [width, lineY]
    point_matrix = np.array([point_1, point_2], np.int32)
    return point_matrix.reshape((-1, 1, 2))
```

U ovom dijelu koda definiramo funkciju koja nam radi matricu koju koristimo u prikazu linija i sa njom definiramo polja koja računaju prolaz vozila. Polja se dijele od -2 do 2 sa tim da nam je -2 up line -1 nam je down line, 1 nam je up limit line i 2 nam je down limit line.

```
def printDetectionLines(slika, line_up, line_down, up_limit, down_limit,
width):
    # Up Line
    line_up_points_cord = getDetectionLinePointsMatrix(line_up, width)
    slika = cv2.polylines(slika, [line_up_points_cord], False,
line_up_color, thickness=2)
    # Down Line
    line_down_points_cord = getDetectionLinePointsMatrix(line_down, width)
    slika = cv2.polylines(slika, [line_down_points_cord], False,
line_down_color, thickness=2)
    # Up Limit Line
    limit_line_up_points_cord = getDetectionLinePointsMatrix(up_limit,
width)
    slika = cv2.polylines(slika, [limit_line_up_points_cord], False, (224,
224, 224), thickness=1)
    # Down Limit Line
    limit_line_down_points_cord = getDetectionLinePointsMatrix(down_limit,
width)
    slika = cv2.polylines(slika, [limit_line_down_points_cord], False,
(224, 224, 224), thickness=1)
```

Ovim blokom koda definiramo atribute linija i stavljamo ih u funkciju „printDetectionLines“, koju kasnije pozivamo u main fajl kako bi ih prikazali.

```
def printOutputText(slika, count_up, count_down, count_auto, count_kamion,
accident_warning_flag):
    cv2.rectangle(slika, (5,5), (200, 140), bg_rect_color, cv2.FILLED)
    str_up = 'Prema Gore: ' + str(count_up)
    str_down = 'Prema Dole: ' + str(count_down)
    str_auto = 'Broj Auta: ' + str(count_auto)
    str_kamion = 'Broj Kamiona: ' + str(count_kamion)
    str_total = 'Ukupan broj vozila: ' +str(count_up + count_down)
  str_accident_warning = Potencijalni sudar: ' + str(accident_warning_flag)
    cv2.putText(slika, str_up, (15, 25), font, 0.45, line_up_color, 1,
cv2.LINE_AA)
    cv2.putText(slika, str_down, (15, 45), font, 0.45, line_down_color, 1,
cv2.LINE_AA)
    cv2.putText(slika, str_auto, (15, 75), font, 0.45, auto_boja, 1,
cv2.LINE_AA)
    cv2.putText(slika, str_kamion, (15, 95), font, 0.45, kamion_boja, 1,
cv2.LINE_AA)
    cv2.putText(slika, str_total, (15, 125), font, 0.45, (255, 255, 255),
1, cv2.LINE_AA)
```

```
cv2.putText(image, str_accident_warning, (15, 155), font,0.45,
accident_warning_color, 1, cv2.LINE_AA)
```

Funkcija „printOutputText“, koju kasnije također pozivamo u main datoteci, ispisuje nam na video stanje prometa. Stanja koja smo definirali su Prolazak vozila prema naprijed, prema nazad, broj automobila koji su prošli, broj kamiona koji su prošli te ukupan broj vozila koja su prošla naprijed i nazad. Također u nastavku im definiramo atribute i poziciju gdje će se nalaziti na video materijalu.

```
def drawVehicleBoundaryRect(slika, bounding_rect_x, bounding_rect_y,
bounding_rect_w, bounding_rect_h, normal_vehicle):
    rect_color = auto_boja if normal_vehicle else kamion_boja

    # crtanje kvadrata
    cv2.rectangle(slika, (bounding_rect_x, bounding_rect_y),
(bounding_rect_x + bounding_rect_w, bounding_rect_y +
bounding_rect_h),rect_color, 2)

    # davanje atributa pravokutnika (vozila)
    label = 'Auto' if normal_vehicle else 'Kamion'
    labelSize, baseLine = cv2.getTextSize(label, font, 0.5, 1)
    cv2.rectangle(slika, (bounding_rect_x -1, bounding_rect_y -
labelSize[1] - 8), (bounding_rect_x + labelSize[0] + 2, bounding_rect_y),
rect_color, cv2.FILLED)
    cv2.putText(slika, label, (bounding_rect_x, bounding_rect_y - 6), font,
0.5, (0, 0, 0), 1, cv2.LINE_AA)
```

Funkcija „drawVehicleBoundaryRect“ nam je funkcija pomoću koje prikazujemo pravokutnik koji obilježava određeno vozilo te njegov atribut. Nju također kasnije pozivamo u main fajl.

```
def drawVehicleCenterPoint(slika, cx, cy):
    # crtanje kruga
    cv2.circle(slika, (cx, cy), 5, (0, 0, 255), -1)
```

Sljedeća funkcija nam je jednostavna funkcija koja nam crta krug te ga postavlja u sredini detektiranog vozila.

```
def showMoveWindow(slika, text, x, y):
    cv2.imshow(text, slika)
    cv2.moveWindow(text, x, y)

def displayImageAndTransformations(slika, imBin, c_mask, o_mask, height,
width):
    showMoveWindow(slika, 'Frame', 0, 0)
    showMoveWindow(imBin, 'Subtraction and Binarization (treshold)',
int(width), 0)
    showMoveWindow(c_mask, 'Closing Morphology', 0, int(height))
    showMoveWindow(o_mask, 'Opening Morphology', int(width), int(height))
```

Zadnje dvije funkcije „showMoveWindow“ i „displayImageAndTransformations“ su dvije povezane funkcije i one nam služe kako bi nam otvorili prozore tj. prikazale videa i sve transformacije koje su se radile u postupku. Njih ćemo pojasniti malo dalje u radu.

5.3. „my_vehicle.py“ Kod

Ovaj dio koda prikazuje nam klasifikaciju vozila i logiku sa kojom smo odradili praćenje vozila. Ona se naknadno upotpunjuje u main datoteci, koja će biti prikazana u [5.4].

Kako bi se započela klasifikacija potrebno je uvesti dvije biblioteke. Biblioteka Time i biblioteka random. Time biblioteka (biblioteka vremena) nam je standardna pythonova biblioteka za vrijeme. U ovom radu ona je korištena kako bi dokumentirali vrijeme i datum prolaženja vozila, zbog toga što se ovakva tehnologija primjenjuje u stvarnom vremenu iz live feed-a[10]. U nastavku je primjer iz pythona kako izgleda ispis praćenja:

```
ID:  1 Prošao [DOLE] u Fri Oct  8 16:37:46 2021
ID:  4 Prošao [DOLE] u Fri Oct  8 16:37:49 2021
ID:  3 Prošao [DOLE] u Fri Oct  8 16:37:49 2021
ID:  6 Prošao [GORE] u Fri Oct  8 16:37:54 2021
ID:  7 Prošao [DOLE] u Fri Oct  8 16:37:57 2021
ID:  9 Prošao [DOLE] u Fri Oct  8 16:38:06 2021
ID:  10 Prošao [GORE] u Fri Oct  8 16:38:09 2021
ID:  12 Prošao [GORE] u Fri Oct  8 16:38:18 2021
ID:  13 Prošao [DOLE] u Fri Oct  8 16:38:30 2021
Ukupni broj prođenih vozila:  9
Video je završio.

Process finished with exit code 0
```

Slika 11. Prikazuje vrijeme u kojem je vozilo detektirano

[10] Live feed – Videomaterijal koji se prenosi u stvarnom vremenu prema centrali za praćenje prometa.

Sljedeća biblioteka koju smo koristili je biblioteka random, točnije funkcija randint iz biblioteke random. To nam je funkcija koja nam vraća nasumične brojeve iz zadane ulazne i izlazne vrijednosti. U nastavku koda radimo klasu sa nazivom vozila i u nj. definiramo atribute koje koristimo u detekciji i klasifikaciji.

```
class Vozila:
    tracks = []
    def __init__(self, i, xi, yi, area, max_age):
        self.i = i
        self.x = xi
        self.y = yi
        self.area = area
        self.tracks = []
        self.R = randint(0, 255)
        self.G = randint(0, 255)
        self.B = randint(0, 255)
        self.done = False
        self.state = '0'
        self.age = 0
        self.max_age = max_age
        self.dir = None
        self.type = 'normal'
```

Nakon što smo postavili atribute za klasu vozila radimo funkcije na koje ćemo se pozivati u nastavku koda.

```
    def getRGB(self):
        return (self.R, self.G, self.B)

    def getTracks(self):
        return self.tracks

    def getId(self):
        return self.i

    def getState(self):
        return self.state

    def getDir(self):
        return self.dir

    def getX(self):
        return self.x

    def getY(self):
        return self.y

    def getArea(self):
        return self.area

    def updateArea(self, area):
        self.area = area
    def getType(self):
        return  self.type
```

```
    def setType(self, type):
        self.type = type

    def updateType(self, treshold):
        if self.area > treshold:
            self.type = 'Kamion'
        else:
            self.type = 'Auto'

    def updateCoords(self, xn, yn):
        self.age = 0
        self.tracks.append([self.x, self.y])
        self.x = xn
        self.y = yn

    def setDone(self):
        self.done = True

    def timedOut(self):
        return self.done
```

Prva logička funkcija koja nam služi za detekciju prometa nam je „going_UP“.

```
def going_UP(self, detection_line):
        if len(self.tracks) >=2:
            if self.state == '0':
                if self.tracks[-1][1] < detection_line and self.tracks[-
2][1] > detection_line:
                    self.state = '1'
                    self.dir = 'up'
                    return True
                else:
                    return False
            else:
                return False
        else:
            return False
```

Ova funkcija nam govori da ako je auto ušlo u zonu detekcije i ako je prošlo od donje limit linije do gornje limit linije biti će detektirano kao da je prošlo prema gore.

```
   def going_DOWN(self, detection_line):
        if len(self.tracks) >=2:
            if self.state == '0':
                if self.tracks[-1][1] > detection_line and self.tracks[-
2][1] < detection_line:
                    self.state = '1'
                    self.dir = 'down'
                    return True
                else:
                    return False
            else:
                return False
        else:
            return False
```

Isto tako samo na suprotnu stranu govori nam funkcija going_DOWN.

Kako bi mogli pratiti vozila koja se pojavljuju radimo funkciju koja će nam postavljati broj pored svakog vozila koji detektira. Nakon što detektira i postavi broj, funkcija pamti broj na kojem je stala te za svako novo vozilo pridružuje novi broj (ID).

```
def age_one(self):
    self.age += 1
    if self.age > self.max_age:
        self.done = True
    return True

def getAge(self):
    return self.age
```

5.4. „main.py“ Kod

Započinjemo main kod sa uvozom biblioteke time koje smo spomenuli u [5.3] i OpenCV biblioteke spomenute u [5]. Kako bi mogli pozvati funkcije koje smo prije spomenuli moramo ih također uvesti sa import funkcijom:

```
import time

import cv2

import my_vozilo as vozilo
import my_ekran as ekran
```

Kako bi si kasnije olakšali pisanje koda sa funkcijom „as“ možemo si skratiti ima fajla sa npr. my_vozilo na vozilo.

U nastavku zadajemo globalne varijable:

```
input_video_path = 'Video1.mp4'
cap = cv2.VideoCapture(input_video_path)

omjer = .45
```

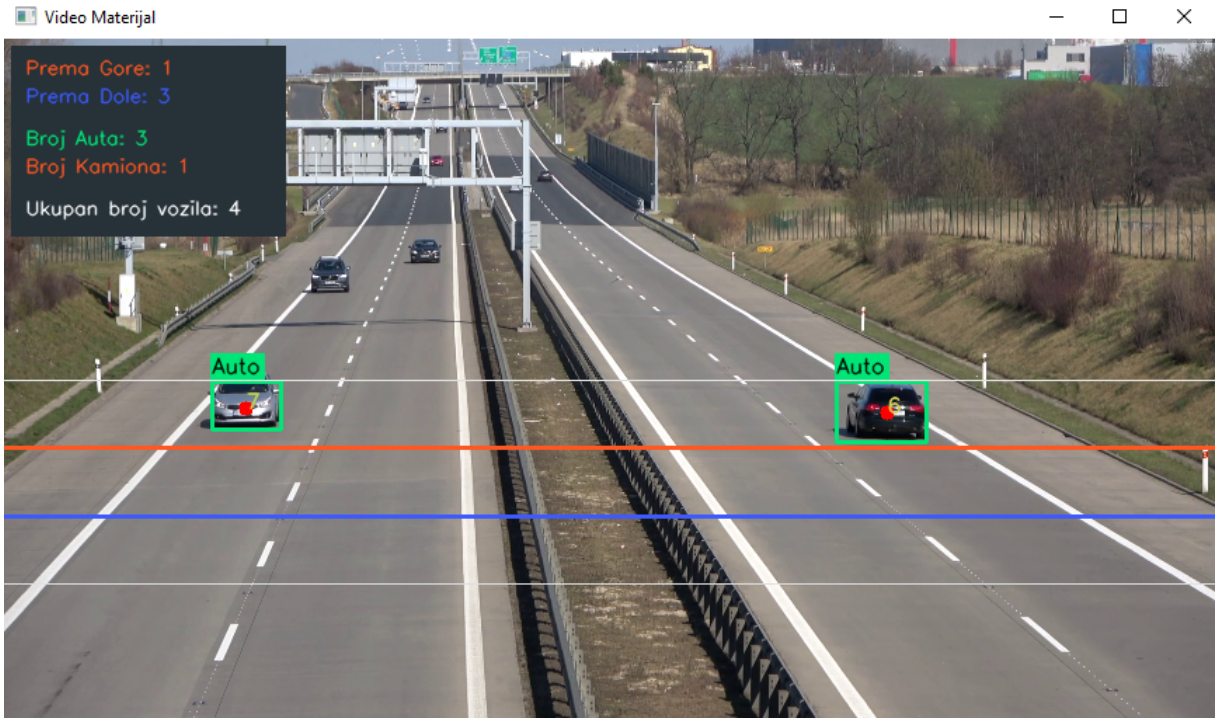
U Python uvozimo video pod nazivom Video1 te ga pomoću funkcije cv2.VideoCapture definiramo kao objekt s kojim kasnije upravljamo. U nastavku si zadajemo omjer koji ćemo detaljnije spomenuti u nastavku.

```
sirina = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)) * omjer
visina = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) * omjer
```

Dalje u kodu definiramo širinu i visinu videa pomoću funkcija cv2.CAP_PROP_FRAME_WIDTH i cv2.CAP_PROP_FRAME_HEIGHT pomnoženu sa omjerom koji smo zadali prije. Mijenjanjem omjera, možemo si prilagođavati veličinu videa prema potrebi.

```
frameArea = visina * sirina
areaTR = frameArea / 400
type_of_vehicle_area_treshold = 10000
```

Sljedećim djelom koda određujemo si količinu prostora u videu koji će biti rezerviran za detekciju prometa. To radimo kako bi smanjili nepotrebne šumove i povećali preciznost samog sustava. U nastavku je prikaz polja za detekciju.



Slika 12. Prikaz polja za detekciju

Kao što možemo primijetiti na slici 11 u lijevoj traci vozila nam se detektiraju oko prve linije. Sve što se nalazi iza prve linije ne smatra se poljem detekcije. Ovakvom metodom smanjujemo količinu procesorske snage koja je potrebna za obradu video materijala te samim time poboljšavamo detekciju. Zadnja linija koda type_of_vehicle_area_treshold nam je broj koji se kasnije javlja u kodu kao broj sa kojim mjerimo veličinu objekta u videu.

Sljedećim linijama koda zadajemo si pozicije linija koje su prikazane videu koje također možemo vidjeti u slici 11.

```
up_limit = int(5 * (visina / 10))
line_up = int(6 * (visina / 10))
line_down = int(7 * (visina / 10))
down_limit = int(8 * (visina / 10))
```

Dalje kod nastavljamo sa funkcijama za obradu videa. Prva nam je:

```
fgbg = cv2.createBackgroundSubtractorMOG2()
```

Ova funkcija iz biblioteke OpenCV-a nam je algoritam koji generira binarnu masku preko videa. Nju kasnije pozivamo u nastavku koda.

Druga obrada videa nam je algoritam kernel ili u doslovnom prijevodu „zrno“. Ovaj algoritam nam dopušta mijenjanje vrijednosti svakog piksela kombinirajući ga s različitim količinama susjednih piksela u fotografiji/videu.

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
```

U nastavku koda radimo listu „my_vozilo“ u kojoj se pohranjuju detektirane vrijednosti i početno stanje detekcije nezgoda. Početna vrijednost u listi svih elemenata nam je nula osim vehicle_id zbog toga što nam ono označava broj vozila krećući od 1.

```
my_vozilo = []
count_up = 0
count_down = 0
count_kamion = 0
count_auto = 0
max_vehicle_age = 5
vehicle_id = 1
accident_warning_flag = False
```

Nakon što smo definirali globalne varijable krećemo sa analizom videa. Prvi dio koji radimo je taj da otvaramo while petlju te postavljamo uvjet da će se petlja vrtiti sve dok traje video materijal. Nakon što video završi u konzoli printamo broj prijeđenih vozila u videu te printamo tekst da je video završio.

```
while(cap.isOpened()):
    ret, frame = cap.read()

    if not ret:
        print("Ukupni broj prođenih vozila: ", (count_up + count_down))
        print("Video je završio.")
        break
```

U nastavku koda pozivamo se na funkciju age_one() iz my_vozila fajla. Kao što smo već rekli u [5.3] ona nam indeksira svako vozilo u prolazu.

```
   for v in my_vozilo:
        v.age_one()
```

Prije početka detaljne predobrade videa smanjujemo rezoluciju videa na onu koja nam odgovara za prikaz na računalu, zbog toga jer je ulazna rezolucija videa veća od rezolucije ekrana na kojem vršimo obradu. Tu pretvorbu izvodimo naredbom :

```
slika = cv2.resize(frame, (0, 0), None, omjer, omjer)
```

Prva predobrada nam je izuzimanje slika iz videa kako bi napravili masku po kojoj možemo raditi promjene.

```
fgmask = fgbg.apply(slika)
```

Sljedeća predobrada nam je binarizacija videa i nju izvodimo sa funkcijom cv2.treshold

```
ret, binarna_slika = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)
```

Sljedeće dvije predobrade videa su nam morfološke operacije zatvaranja i otvaranja. To znači da prilikom operacije otvaranja uklanjamo buku u videu tj. uklanjamo manje oblike smetnje u videu. Dok operacijom zatvaranja rješavamo se smetnji u objektima koji se detektiraju u obliku malih crnih praznina.

```
o_slika = cv2.morphologyEx(binarna_slika, cv2.MORPH_OPEN, kernel)
c_slika = cv2.morphologyEx(o_slika, cv2.MORPH_CLOSE, kernel)
```

Slika 13. Prikaz morfološke operacije otvaranja i zatvaranja

U slici 12 vidimo kako nam se morfološka operacija otvaranja i zatvaranja obavila u potpunosti kod manjeg vozila, dok kod većeg vozila potrebno je još malo vremena kako bi se obavila potpuna operacija.

Nakon predobrade videa pišemo dio koda koji će nam pronalaziti konture unutar videa.

```
all_contours, hierarchy = cv2.findContours(c_slika, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

normalno_vozilo = True
```

Pri završetku funkcije age_one postavljamo varijablu normalno_vozilo koja nam služi kao oznaka koja će nam poslužiti u razlučivanju između kamiona i automobila.

U nastavku koda krećemo sa for petljom u kojoj definiramo da varijabla area[11] predstavlja konture u videu i da u slučaju da je „area“ veća od zadane vrijednosti areaTR, što nam predstavlja područje praćenja, izvodimo dio koda koji nam traži centar od detektirane konture.

Nakon te operacije stavljamo novu varijablu new = true koja nam kasnije radi tako da radi nove kategorije prilikom detekcije, u našem slučaju auto i kamion.

```
    for contour in all_contours:
        area = cv2.contourArea(contour)
        if area > areaTR:

            m = cv2.moments(contour)
            cx = int(m['m10'] / m['m00'])
            cy = int(m['m01'] / m['m00'])

            bounding_rect_x, bounding_rect_y, bounding_rect_w,
bounding_rect_h = cv2.boundingRect(contour)

            new = True
```

[11] Engl. Area = područje

Kod dalje nastavljamo sa if funkcijom u kojoj se pitamo jesu li konture unutar zone detekcije i sa for petljom provjeravamo je li već detektirano vozilo upisano u listu. Nakon provjere slijedi nam provjera/detekcija nezgoda. Kod provjerava je li vozilo u stanju mirovanja te ako je šalje nam signal kako je riječ o mogućoj nezgodi.

```
if cy in range (up_limit, down_limit):
                for v in my_vozilo:
                    if abs(bounding_rect_x - v.getX()) <= bounding_rect_w
and abs(bounding_rect_y - v.getY()) <= bounding_rect_h:
                        new = False

                        movement_cx = abs(v.getX() - cx)
                        movement_cy = abs(v.getY() - cy)
                        if movement_cx == 0  and movement_cy == 0:
                            accident_warning_flag = True


                        v.updateCoords(cx, cy)
                        v.updateArea(area)
                        v.updateType(type_of_vehicle_area_treshold)
```

Nakon provjere je li objekt upisan u listu i provjere potencijalne nezgode kreće kod detekcije vozila te razvrstavanja. U if funkciji pozivamo se na funkcije iz „my_vozilo“ koda i pitamo je li vozilo prošlo gornju liniju. Ako je prošlo gornju liniju upisujemo u gornji brojač +1 te pitamo dalje da upiše + 1 ako je prošao kamion ili +1 ako je prošao auto. Taj podatak znamo jer smo na početku zadali da nam je „type_of_vehicle_area_threshold = 10000“ i on nam razlikuje auto od kamiona po broju piksela koji se nalazi u konturama.

```
if v.going_UP(line_up) == True:
                            count_up += 1
                            count_kamion += 1 if v.type == 'Kamion' else 0
                            count_auto += 1 if v.type == 'Auto' else 0
                            print("ID: ", v.getId(), 'Prošao [GORE] u',
time.strftime("%c"))
                        elif v.going_DOWN(line_down) == True:
                            count_down += 1
                            count_kamion += 1 if v.type == 'Kamion' else 0
                            count_auto += 1 if v.type == 'Auto' else 0
                            print("ID: ", v.getId(), 'Prošao [DOLE] u',
time.strftime("%c"))
                        break
```

Sljedeći if blok koda govori nam kada trebamo prestati pratiti vozilo ako je prošlo gornju ili donju liniju.

```
if v.getState() == '1':
      if v.getDir() == 'down' and v.getY() > down_limit:
            v.setDone()
      elif v.getDir() == 'up' and v.getY() < up_limit:
            v.setDone()
```

Sljedeći if blok nam zatvara for petlju sa pitanjem je li novo vozilo koje je prošlo upisano u listu. Ako nije upisuje se u listu, ako je već u listi kod se nastavlja dalje.

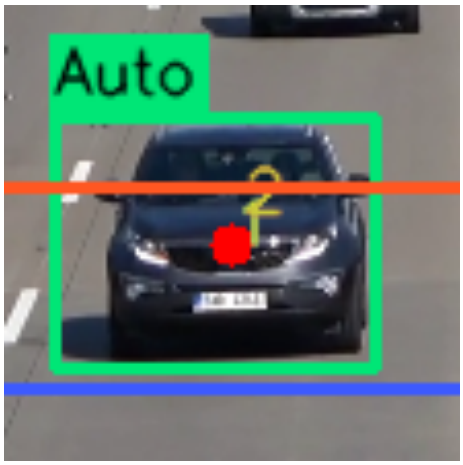
```
if new == True:
                    new_Vehicle = vozilo.Vozila(vehicle_id, cx, cy, area,
max_vehicle_age)
                    my_vozilo.append(new_Vehicle)
                    vehicle_id += 1
```

Sljedeći if blok nam zatvara početnu if petlju za detekciju vozila. Ona pita je li kontura veća od zadanog proja piksela. U slučaju da je dobivamo rezultat kamiona. Ovaj if blok nam je zapravo razlučivanje vozila između kamiona i automobila.

```
if area > type_of_vehicle_area_treshold:
                normalno_vozilo = False
```

Sljedeće linije koda pozivaju nam iz my_ekran koda kod za prikaz pravokutnika te srednje točke.

```
my_ekran.drawVehicleCenterPoint(slika, cx, cy)
            my_ekran.drawVehicleBoundaryRect(slika, bounding_rect_x,
bounding_rect_y, bounding_rect_w, bounding_rect_h, normalno_vozilo)
```



Slika 14. Prikaz pravokutnika oko vozila te srednje točke

Kako bi prikazali ID od vozila koristimo ovaj blok koda:

```
for v in my_vozilo:

    if not v.timedOut():
        my_ekran.printVehicleId(slika, v)
    else:
        index = my_vozilo.index(v)
        my_vozilo.pop(index)
        del v
```

Sljedeće tri linije koda nam zapravo ispisuju podatke na video. Ispisuju se linije detekcije, text u gornjem lijevom kutu videa te transformacije/predobrade koje smo definirali u kodu.

```
my_ekran.printDetectionLines(slika, line_up, line_down, up_limit,
down_limit, sirina)

my_ekran.printOutputText(slika, count_up, count_down,
count_auto,count_kamion, accident_warning_flag)
accident_warning_flag = False

my_ekran.displayImageAndTransformations(slika, binarna_slika, c_slika,
o_slika, sirina, visina)
```

Na kraju main koda postavljamo kratku if naredbu koja nam daje kratku pauzu u milisekundama između svake slike u videu kako bi mogli zaustaviti video. Postavljen je razmak od 30 milisekundi i tipka „q“ na tipkovnici.

```
if cv2.waitKey(30) & 0xff == ord('q'):
        break
```

Na kraju ostale su nam još dvije naredbe a to su :

```
cap.release()
cv2.destroyAllWindows()
```

One nam služe kako bi se zaustavili svi procesi i zatvorili svi prozori koje je program otvorio.

6. Pogreška u detekciji vozila

U ovom radu postoje pogreške u detekciji vozila i one se dešavaju ovisno jačini hardvera na kojem se rade operacije/izvodi kod. S obzirom na tu činjenicu ovdje su pogreške u detekciji sa hardverom koji je korišten u realizaciji ovog rada.

Ukupan broj vozila koja su prošla u obje trake	Ukupan broj detektiranih vozila u obje trake	Broj pogrešnih detekcija iz okoline (Ne utječu na detekciju vozila)	Ukupan broj krivo detektiranih vozila	Učinkovitost
11	9	0	0	81.8%

Tablica 1. Prikaz učinkovitosti detekcije ovoga sustava

Pošto nam za ovaj video isječak greške iz okoline ne utječu na detekciju vozila, nemamo krivo detektirana vozila i ovisno hardveru koji je korišten u izračunu matematičkih operacija možemo reći da je učinkovitost ovog sustava 81,8% za trenutni hardver.

7. Zaključak

Premda već postoje sustavi za detekciju i praćenje prometa, zamisao ovog rada bila je na pojednostavljen način prikazati i implementirati sustav za detekciju i praćenje prometa. Kako bi prikazali sustav korišten je Python programski jezik i njegove biblioteke te OpenCV biblioteka. Praćenje prometa izvedeno je uz pomoć algoritama iz biblioteke OpenCV kojim smo mijenjali postavke video materijala tj. radili konverzije i poboljšanja video materijala koje su nam dale mogućnost detektiranja i klasifikacije vozila. Uz pomoć pythona napravljena je klasifikacija vozila i logika praćenja kojom su dobiveni rezultati ovog rada. Primjena ovakvog sustava je iznimno široka zbog toga što je Python programski jezik visoke razine i njegova primjena je svugdje gdje postoji neki oblik koda, a biblioteka OpenCV nije ograničena na određene objekte već je moguće detektirati bilo što ovisno o potrebi.

Bez obzira na to što je ovaj rad pojednostavljeni prikaz detekcije i praćenja programa postoji više načina kako bi mogli nadograditi/poboljšati ovaj rad kao npr. dublja klasifikacija vozila, još precizniji sustav detekcije vozila itd.

U ovom radu uspješno smo prikazali detekciju i praćenje prometa i ustanovili nedostatke i moguća poboljšanja što ovaj rad privodi kraju.

Literatura

1. Novosel J. , Zagreb, "Sustav računalnog vida za automatsko prepoznavanje vozila u svrhu nadzora prometa", 2011

2. Mandžuka S, Zagreb, "INTELLIGENT TRANSPORT SYSTEM Selected Lectures" 2015

3. Kurdi H, April 2014. "Review of Closed Circuit Television (CCTV) Techniques for Vehicles Traffic Management", International Journal of Computer Science & Information Technology (IJCSIT) Vol 6, No 2

4. Husain A, Maity T, Yadav R, 2019, "Vehicle detection in intelligent transport system under a hazy environment: a survey", Dostupno na [https://www.researchgate.net/publication/346715449_IET_Image_Processing_Vehicle_detection_in_intelligent_transport_system_under_a_hazy_environment_a_survey]

5. Horvat B, Matić I, Rijeka, "INTELIGENTNI SUSTAVI UPRAVLJANJA PROMETOM", 2010

6. Faletar M, Osijek, "INTELIGENTNI TRANSPORTNI SUSTAVI", 2020

7. Barron, J., Fleet, D., Beauchemin, S, 1994, .: ‘Performance of optical flow techniques’, Int. J. Comput. Vis.

8. Aslani S, Mahdavi-Nasab H, 2013, "Optical Flow Based Moving Object Detection and Tracking for Traffic Surveillance" International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:7, No:9,

9. „Python Project Tutorial – Vehicle Detection And Counting using OpenCV | Vehicle Counting using OpenCV, 2021, Dostupno na : [https://www.youtube.com/watch?v=6kZftXunlTY]

10. „Object Tracking with Opencv and Python“, 2021, Dostupno na : [https://pysource.com/2021/01/28/object-tracking-with-opencv-and-python/]

11. Mufti N, Shah S, 2021, „Automatic Number Plate Recognition: A Detailed Survey of Relevant Algorihms“, Dostupno mna stranici : [https://core.ac.uk/download/pdf/25846945.pdf]

12. Pavani T, Mohan DVR, (2019), „Number Plate Recognition by using Open CV – Python, International Research Journal of Engineering and Technology“, Dostupno na : [https://www.irjet.net/archives/V6/i3/IRJET-V6I31275.pdf]

13. Wolfewicz A, (2021), "A begginer's guide to how machines learn" Raspoloživo na [https://levity.ai/blog/how-do-machines-learn]

Popis slika

Popis tablica