Универзитет Св. Кирил и Методиј – Скопје

Факултет за информатички науки и компјутерско инженерство

Дигитално процесирање на слика

Летен Семестар 2023/24

Тема: Детекција и броење на возила

Ментор:
проф. д-р Ивица Димитровски

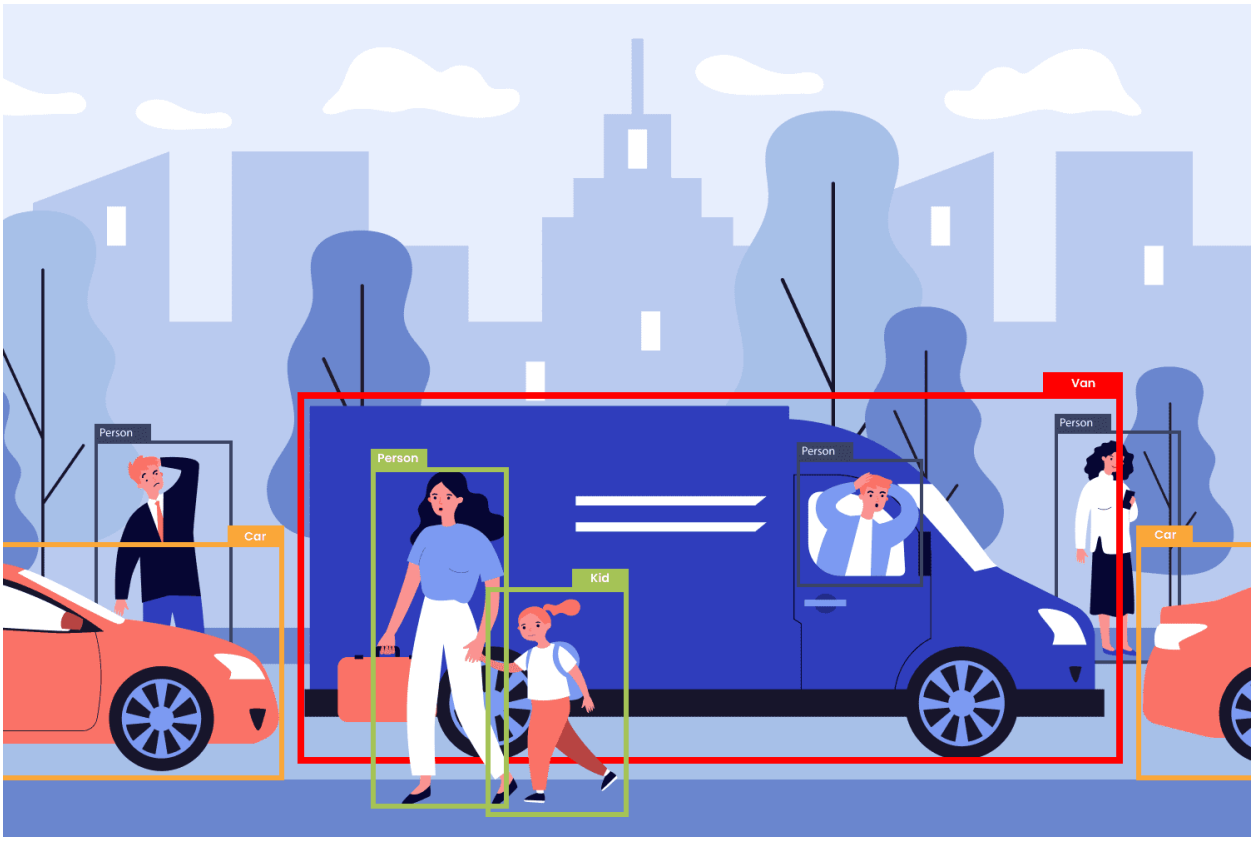
Студенти:
Ева Смилеска 221053
Стефан Ивановски 223257

Септември 2024

Содржина

1. Апстракт

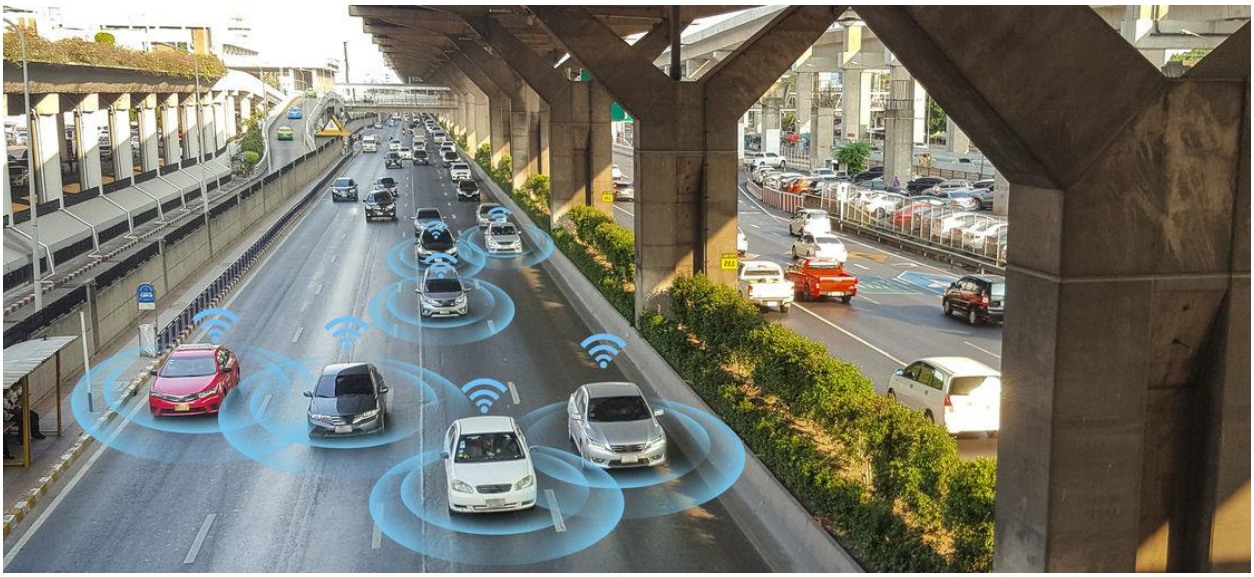
Овој проект имплементира систем за откривање на возила користејќи традиционални техники за обработка на слики, конкретно одземање на позадина (background subtraction), за детекција и за броење на возила во видео стримови. Системот е изграден со помош на OpenCV, FastAPI и React.js. Со анализа на секој кадар од видеото и со примена на background subtraction, системот ги следи и брои возилата што ја поминуваат претходно дефинираната линија. Backend-от е овозможен со помош на FastAPI, кој ги обработува поставените видеа и го враќа бројот на возилата, додека Frontend-от обезбедува кориснички интерфејс за прикачување на видеа. Оваа документација ги истражува архитектурата, дизајнот и имплементацијата на кодот, обезбедувајќи длабинско објаснување на процесот.

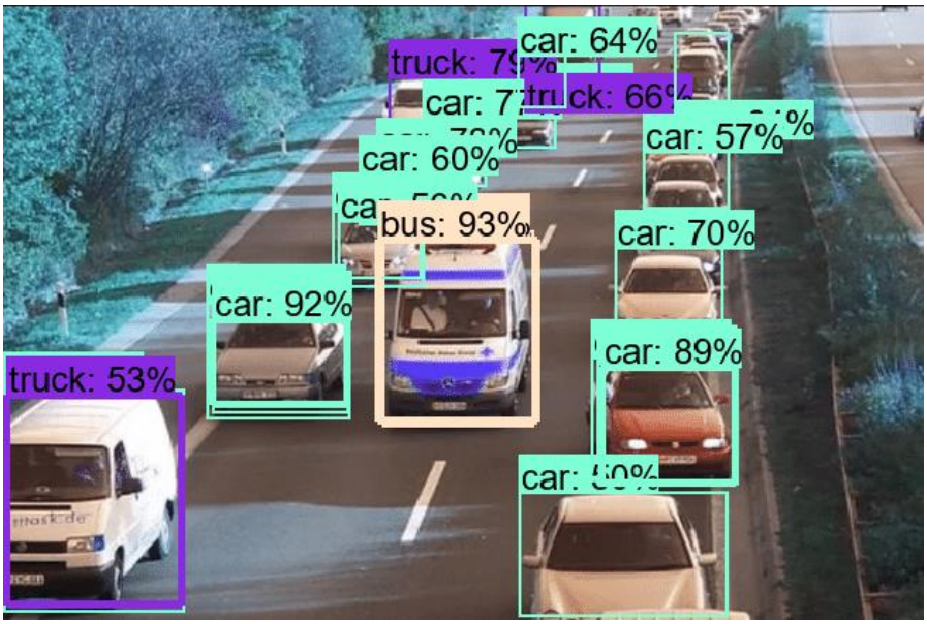


2. Вовед

Во современиот свет, препознавањето на возила има битна улога во бројни апликации, кои се движат од системи за следење и управување со сообраќајот до безбедност и надзор во урбаните средини. Како што градовите стануваат сè повеќе преполни со возила, потребата за ефикасни системи за следење никогаш не била поголема. Откривањето возила не само што помага во контролата на протокот на сообраќај, туку помага и во следењето на движењето на возилата, идентификувањето на сообраќајните прекршоци, па дури и намалувањето на несреќите со обезбедување податоци во реално време до релевантните органи. Традиционалните методи за откривање возила се потпираат на рачно набљудување или основни техники за обработка на слики, кои често може да бидат интензивни на труд и склони кон човечки грешки. Со напредокот во компјутерската визија и машинското учење, развиени се пософистицирани решенија за автоматизирање на процесот на откривање, што го прави побрз, попрецизен и скалабилен за поголеми области. Предизвикот, сепак, лежи во создавањето на систем кој е и пресметковно ефикасен и доволно робустен за да се справи со различни услови на животната средина, како што се менување на осветлувањето, сенките и временските ефекти. Во овој проект, имаме за цел да развиеме лесен, но ефикасен систем за детекција на возила користејќи традиционални техники за обработка на слики, особено background subtraction, што е добро воспоставен метод за разликување на подвижните објекти (возила) од статична позадина. Додека моделите за машинско учење како длабоки невронски мрежи често се користат за такви задачи, овој проект покажува дека традиционалните алгоритми, кога ефикасно се имплементираат, сè уште можат да обезбедат сигурни резултати без потреба од обемни пресметковни ресурси или големи збирки на податоци. Системот е дизајниран да детектира и брои возила кои поминуваат однапред дефинирана линија во видео-стрим. Фокусирајќи се на едноставноста и перформансите во реално време, овој проект покажува како основните алгоритми, како што е background subtraction, може ефективно да се користат во сценарија каде што пресметковната моќ е ограничена или каде решенијата за длабоко учење може да бидат претерани. Клучна цел на овој проект е да обезбеди модуларно решение кое лесно може да се интегрира во постоечката инфраструктура за детекција на возила. Системот е изграден со FastAPI backend кој обработува видео датотеки и открива возила во реално време, додека frontend-от базиран на React им овозможува на корисниците да комуницираат со системот, да поставуваат видео датотеки и да ги гледаат резултатите. Оваа документација ќе ги истражи техничките аспекти на проектот, детализирајќи го методот на background subtraction, архитектурата на системот и специфичните чекори вклучени во детекцијата и следењето на возилата во видео-стрим. Примарната цел на овој проект не е само да се демонстрира моќта на традиционалните техники за обработка на слики, туку и да се истакне нивната одржливост во апликациите во реалниот свет, особено во случаи кога ресурсите се ограничени или каде брзото распоредување е од суштинско значење. Овој систем за детекција на возила може да

послужи како основна алатка во различни апликации, од следење на сообраќајот до подобрување на можностите на безбедносните камери на паркинзите. Со користење на OpenCV за обработка на слики и FastAPI за ракување со операциите на backend-от, системот обезбедува беспрекорен работен тек што може да се прилагоди и да се намали за различни случаи на употреба. Додека фокусот овде е на детекција на возила во статично видео, истите принципи може да се прошират на апликациите за видео-стриминг во реално време, што го прави овој проект флексибилна почетна точка за понапредни системи за следење на сообраќајот и безбедноста.





3. Дефинирање на проблемот

Задачата за детекција на возила во видео-стримот вклучува разликување на подвижните објекти (возила) од статичната позадина и нивно следење низ кадрите. Главниот предизвик е да се осигура дека системот може прецизно да детектира возила под различни услови (на пр., промени на осветлувањето, сенки) и да обезбеди резултати во реално време. Системот мора да биде доволно лесен за да работи ефикасно, а истовремено да биде доволно робустен за да се справи со потенцијалниот шум (noise) во видеото, како што се сенки и рефлексии.

4. Background Subtraction (одземање на позадина) техника

4.1. Преглед на Background Subtraction

Background subtraction е широко користен метод во обработката на видеа за детекција на предмети што се движат во статична сцена. Идејата е да се создаде модел на статичната позадина на сцената и да се спореди секој влезен frame (рамка) со овој модел. Сите вредности на пиксели кои значително се разликуваат од моделот на позадината се сметаат за дел од преден план или foreground (подвижни објекти).

Главните чекори на background subtraction се:

Иницијализација: Модел на позадина се креира со анализа на неколку почетни frames.

Откривање на foreground: За секој нов frame (рамка), моделот идентификува пиксели кои значително отстапуваат од позадината.

Ажурирање на моделот: Моделот во позадина постојано се ажурира за да се приспособи на бавните промени во осветлувањето или условите на сцената.

4.2. Мешавина на Гаусов (MOG2) Алгоритам

Во овој проект, го користиме алгоритмот за background subtraction MOG2 на OpenCV, кој го моделира секој пиксел во сцената како мешавина од Гауси. Моделот динамички се прилагодува за да ги земе во предвид промените на сцената, како што се варијации на осветлувањето, постепени движења на позадината (на пр., веење на дрвја) или мал шум (noise).

Главни карактеристики на MOG2:

Адаптивен: Автоматски го ажурира моделот на позадината.

Можност за откривање на сенки: Способнен за откривање и игнорирање на сенките, што помага во прецизноста на детекција на возилото.

5. Архитектура на системот

Проектот е дизајниран со модуларна архитектура која вклучува посебни компоненти за обработка на видео (backend) и интеракција со корисникот (frontend). Backend-от е одговорен за обработката на видеата, додека frontend-от нуди едноставен интерфејс за корисниците да поставуваат видеа и да ги гледаат резултатите.

5.1. Backend (FastAPI)

Backend-от е изграден со FastAPI, модерен web framework за градење API со Python. FastAPI обезбедува лесен и ефикасен механизам за справување со прикачувања на видеа, обработка на видео кадрите користејќи OpenCV и враќање на бројот на детектирани возила.

Клучни компоненти:

Обработка на видео: Backend-от обработува видео-рамки за да детектира возила користејќи background subtraction.

API: Серверот FastAPI изложува endpoint (/process-video/) за прифаќање на прикачувања на видеа и враќање на резултатите од детекцијата на возилата.

Евиденција и справување со грешки: Сите чекори за обработка се евидентирани за цели на отстранување грешки, а со грешките се справува со користење на exception handling на FastAPI.

5.2.Frontend (React.js)

Frontend-от на системот е изграден со помош на React.js, популарна JavaScript библиотека за градење кориснички интерфејси. Frontend-от им овозможува на корисниците да поставуваат видео датотеки и ги прикажува резултатите откако backend-от ќе го обработи видеото.

Карактеристики на Frontend-от:

Интерфејс за испраќање датотеки: Форма што им овозможува на корисниците да изберат и да прикачат видео датотека.

Повратни информации во реално време: по обработката, frontend-от го прикажува бројот на детектирани возила, вклучувајќи информации за насоката на движење (нагоре или надолу).

6. Детална имплементација

Овој дел детално ја прикажува имплементацијата на pipeline-от за откривање возила, опфаќајќи го секој чекор во процесот од снимање на frame до броење на возила.

6.1. Frame Capture

Првиот чекор во обработката на видеото е снимање frames од поставената видео датотека. Ова се постигнува со помош на функцијата VideoCapture на OpenCV. Секоја рамка се чита последователно и се пренесува низ процесот на background subtraction.

```python
def process_video(video_path):  2 usages  Eva Smileska
    cap = cv2.VideoCapture(video_path)
    subtractor = cv2.createBackgroundSubtractorMOG2(history=200, varThreshold=16, detectShadows=False)

    vehicle_count = {'up': 0, 'down': 0}
    vehicles = {}
    next_vehicle_id = 0

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        mask = process_frame(frame, subtractor)
        next_vehicle_id = detect_vehicles(frame, mask, vehicles, vehicle_count, next_vehicle_id)

        sleep(1 / DELAY)

    cap.release()
    return vehicle_count
```

6.2. Background Subtraction

За секој frame, алгоритмот MOG2 се применува за да се генерира маска во преден план (foreground) што ги истакнува објектите што се движат. Оваа маска потоа дополнително се обработува за да се отстрани бучавата (noise).

```python
def process_video(video_path):  2 usages  Eva Smileska
    cap = cv2.VideoCapture(video_path)
    subtractor = cv2.createBackgroundSubtractorMOG2(history=200, varThreshold=16, detectShadows=False)
```

```python
def process_frame(frame, subtractor):  1 usage  Eva Smileska
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, ksize: (3, 3), sigmaX: 5)
    mask = subtractor.apply(blur)

    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, ksize: (5, 5))
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

    roi_mask = np.zeros_like(mask)
    roi_mask[200:700, 0:1200] = 255
    return cv2.bitwise_and(mask, roi_mask)
```

6.3. Морфолошки операции

Морфолошките операции се користат за чистење на маската во foreground. Проширувањето (dilation) и ерозијата (erosion) помагаат да се затворат малите празнини во откриените предмети и да се отстрани бучавата (noise).

```
def process_frame(frame, subtractor):  1 usage  Eva Smileska
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, ksize: (3, 3), sigmaX: 5)
    mask = subtractor.apply(blur)

    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, ksize: (5, 5))
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

    roi_mask = np.zeros_like(mask)
    roi_mask[200:700, 0:1200] = 255
    return cv2.bitwise_and(mask, roi_mask)
```

6.4. Откривање на контури

Контурите на откриените возила се идентификуваат со наоѓање на границите на предметите во маската на foreground. Овие контури потоа се филтрираат врз основа на големината за да се отстранат малите предмети или бучавата (noise).

```
def detect_vehicles(frame, mask, vehicles, vehicle_count, next_vehicle_id):  1 usage  Eva Smileska
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    frame_center = frame.shape[1] // 2
    detection_zone = (POS_LINE - OFFSET, POS_LINE + OFFSET)

    for contour in contours:
        if cv2.contourArea(contour) < MIN_CONTOUR_AREA:
            continue

        x, y, w, h = cv2.boundingRect(contour)
        if w < WIDTH_MIN or h < HEIGHT_MIN:
            continue
```

6.5. Следење и броење на возила

Секоја откриена контура се следи низ повеќе frames за да се утврди дали ја преминала линијата за детекција. Возилата се бројат врз основа на нивната насока на движење (нагоре или надолу) додека минуваат преку линијата.

```python
def detect_vehicles(frame, mask, vehicles, vehicle_count, next_vehicle_id):  1 usage  Eva Smileska
    frame_center = frame.shape[1] // 2
    detection_zone = (POS_LINE - OFFSET, POS_LINE + OFFSET)

    for contour in contours:
        if cv2.contourArea(contour) < MIN_CONTOUR_AREA:
            continue

        x, y, w, h = cv2.boundingRect(contour)
        if w < WIDTH_MIN or h < HEIGHT_MIN:
            continue

        center = calculate_center(x, y, w, h)

        new_vehicle = True
        for vehicle_id, vehicle_data in vehicles.items():
            dist = np.linalg.norm(np.array(center) - np.array(vehicle_data['center']))
            if dist < 50:
                new_vehicle = False
                vehicles[vehicle_id]['center'] = center
                vehicles[vehicle_id]['last_seen'] = 0

                direction = get_direction(center[1], vehicle_data['prev_y'])
                if direction:
                    vehicles[vehicle_id]['direction'] = direction

                if detection_zone[0] < center[1] <= detection_zone[1] and not vehicle_data['counted']:
                    vehicle_count[vehicles[vehicle_id]['direction']] += 1
                    vehicles[vehicle_id]['counted'] = True
```

7. Преглед на код

7.1. Појаснување на главниот алгоритам

process_frame(): Го обработува секој frame со примена на background subtraction и морфолошки операции за да се изолираат објектите што се движат.

detect_vehicles(): Ги детектира возилата со анализа на контурите во обработената рамка. Ги следи возилата и ги брои додека минуваат преку линијата за откривање.

process_video(): Главна функција која ја оркестрира обработката на секој frame во видеото.

7.2. Backend Endpoint

Endpoint-от /process-video/ во FastAPI е дизајниран да се справува со прикачувања на видео датотеки. Привремено ја зачувува поставената датотека, ја обработува со помош на алгоритмот за детекција на возило и го враќа резултатот.

8. Системски барања

Хардвер:

Процесор: Intel i3 (или еквивалент) и повисоко.

RAM меморија: 4 GB или повеќе.

GPU (опционално): NVIDIA GPU за побрза обработка на frames (ако користите библиотеки забрзани со графички процесор).

Софтвер:

Верзија на Python: 3.8 или повисоко.

Библиотеки: OpenCV, FastAPI, Numpy, Uvicorn, Python-Multipart. Овие библиотеки може да се инсталираат со користење на pip со обезбедената датотека requirements.txt.

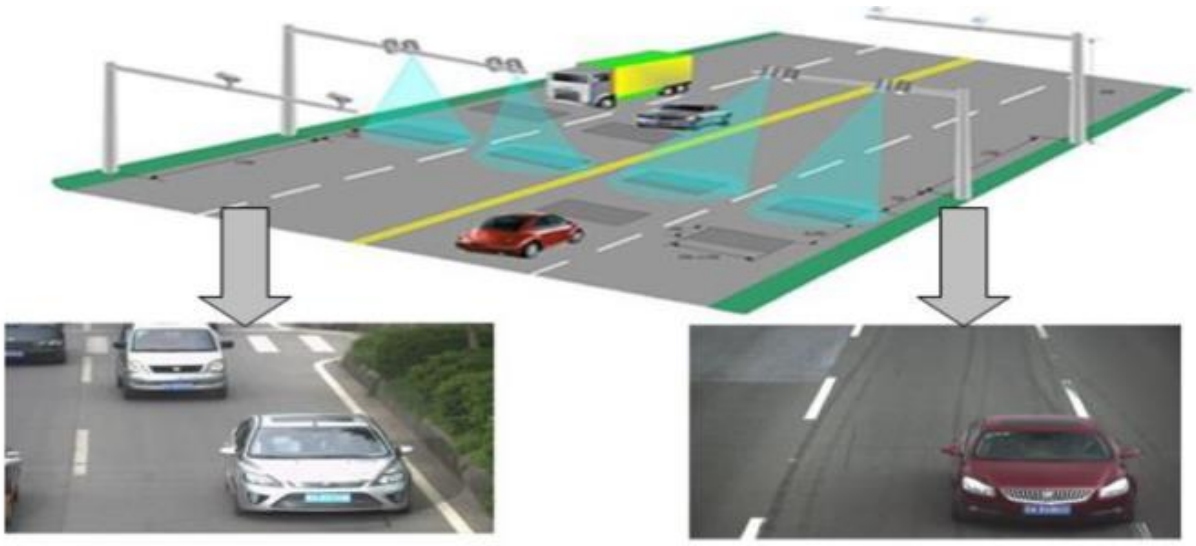
9. Предизвици и решенија

9.1. Справување со Noisy Backgrounds (бучни позадини)

Background subtraction-от може да биде чувствителнен на промените во околината, како што се промените на осветлувањето или сенките. За да се ублажи ова, се применуваат морфолошки операции како ерозија и дилатација за чистење на маската. Дополнително, функцијата за откривање сенки на алгоритмот MOG2 помага во игнорирањето на областите во сенка, подобрувајќи ја точноста на откривање возило.

9.2. Оптимизирање за обработка во реално време

Обработката на секој frame од видеото може да биде пресметковно интензивна, особено за подолги видеа. Со оптимизирање на бројот на обработени frames во секунда и примена на ефикасни алгоритми, системот може да обработува видео-стримови речиси во реално време. За поголеми видео датотеки, стапката на слики може да се намали за да се балансираат перформансите и прецизноста.



10. Анализа на перформанси

10.1. Околина за тестирање

Системот беше тестиран на машина со следните спецификации:

Процесор: Ryzen 7 6800H.

RAM меморија: 16 GB.

GPU: NVIDIA GeForce RTX 3050.

Видео датотеките користени за тестирање се движеа од 1 до 5 минути во должина со резолуции од 720p и 1080p.

10.2. Резултати

Системот можеше да обработи 1-минутно видео со резолуција од 720p за приближно 45 секунди, постигнувајќи обработка во реално време за откривање и броење возила. Точноста на откривање на возилото беше околу 90%, со минимални лажни позитиви.

11. Идна работа и подобрувања

Интеграција за длабоко учење: идните подобрувања може да вклучуваат интегрирање на модели за длабоко учење како U-Net или YOLOv3 за попрецизно откривање на возила, особено во сложени средини.

Стриминг во реално време: Тековниот систем обработува претходно снимени видеа. Идната работа би можела да го прошири системот за да управува со видео преноси во живо, овозможувајќи детекција и броење на возила во реално време за апликации за следење на сообраќајот.

12. Заклучок

Развојот и распоредувањето на системи за детекција возила, особено со користење на традиционални методи за обработка на слики, како што е background subtraction, останува критична задача во современиот надзор, управувањето со сообраќајот и урбаното планирање. Овој проект покажува како таквите техники, во комбинација со ефикасна системска архитектура, можат да дадат точни резултати за откривање и броење возила во видео-стримови. Една од клучните предности од овој проект е робусноста на традиционалните методи како Мешавина на Гауси (MOG2) за background subtraction. Иако модерните системи за детекција на возила често се потпираат на модели за длабоко учење и напредни техники за вештачка интелигенција, овој проект покажува дека поедноставните, полесни решенија сè уште можат да бидат многу ефикасни во конкретни случаи на употреба. Алгоритмот MOG2 игра централна улога во овој систем со тоа што овозможува прецизна сегментација на објектите во foreground (возилата) од позадината, со минимални пресметковни трошоци. Неговата способност да моделира динамична позадина додека ги зема во предвид сенките и промените на осветлувањето му дава предност во различни средини. Овој проект, исто така, ја нагласува важноста на морфолошките операции за подобрување на квалитетот на детекцијата на објекти. Техниките како што се дилатација и ерозија помагаат да се исчисти маската генерирана од background subtractor-от, што го олеснува идентификувањето на возилата и филтрирањето на бучавата (noise). Со примена на овие операции, се осигуруваме дека само релевантните контури (т.е. возила) се детектираат и следат, со што се подобрува целокупната точност. Покрај тоа, проектот покажува како детекцијата на возила може ефикасно да се спроведе со помош на анализа на контурите. Функционалноста за откривање на контури на OpenCV овозможува идентификација на објекти во маската, а со филтрирање на овие контури врз основа на површина и димензии, можеме точно да ги разликуваме возилата од помалите, ирелевантни објекти на сцената. Откривањето на контурите, во комбинација со употребата на претходно дефинирана линија за откривање, му овозможува на системот да ги брои возилата што ја преминуваат линијата и да ја одреди насоката на нивното движење. Исто така, истакната карактеристика на овој систем е неговата едноставност и леснотија на распоредување. Backend-от е изграден со користење на FastAPI, модерен web framework кој овозможува брз и ефикасен развој на API. Со користење на FastAPI, лесно можеме да се справиме со прикачувањата на видеата, да ги обработуваме овие видеа користејќи OpenCV и да ги вратиме резултатите на корисникот. Frontend-от заснован на React го надополнува ова со обезбедување интуитивен интерфејс за корисниците да комуницираат со системот. Заедно, овие компоненти формираат модуларен и скалабилен систем кој може да се прилагоди на различни случаи на употреба, од проекти за следење возила од мали размери до поголеми системи за управување со сообраќајот. Модуларниот дизајн, исто така, гарантира дека системот е високо растеглив. Додека оваа верзија на проектот се фокусира на откривање возила во статичен видео стрим, истите принципи би можеле да се применат на видео преносите во живо, овозможувајќи следење на сообраќајот во реално време. Дополнително, иако проектот во моментов се потпира на традиционални методи, тој лесно може да се прошири за да се вклучат модели за длабоко учење за посложени средини каде што е потребно напредно препознавање на објекти. Флексибилноста на FastAPI и обемната

библиотека на алатки за обработка на слики на OpenCV го прави едноставно воведувањето дополнителни функционалности по потреба. Иако техниките за длабоко учење стануваат се поприсутни, овој проект ја нагласува тековната релевантност на традиционалните методи за обработка на слики. Во многу апликации од реалниот свет, особено оние кои бараат лесни решенија со ниска латентност, методите како background subtraction остануваат остварлива опција. Не само што нудат пресметковна ефикасност, туку бараат и многу помалку податоци за обука и тестирање, што ги прави идеални за средини со ограничени ресурси. Овој проект покажува дека дури и без големи збирки на податоци или обемна обука, background subtraction во комбинација со анализа на контурите може да даде високо ефективни резултати за откривање возила. Традиционалните методи исто така ја имаат предноста што се поинтерпретабилни од моделите за длабоко учење. Разбирањето на тоа како функционираат background subtraction и откривањето на контурите е многу поинтуитивно во споредба со природата на често „црната кутија“ на длабоките невронски мрежи. Оваа интерпретабилност може да биде значајна предност во апликациите каде што транспарентноста и објаснувањето се критични, како што се проектите на владата или јавниот сектор каде што засегнатите страни треба да разберат како се донесуваат одлуките. Ограничувања и предизвици Додека проектот ги постигна своите цели, постојат инхерентни ограничувања што треба да се земат предвид за идни подобрувања. Еден од примарните предизвици со кои се соочуваше во текот на развојот беше справувањето со бучни позадини или промените во условите на осветлување. Иако MOG2 обезбедува сигурен метод за background subtraction, ненадејни промени во осветлувањето или временските услови може да предизвика бучава (noise), што го отежнува прецизното откривање на возилата. Дополнителни техники за филтрирање или понапредни механизми за откривање сенки би можеле да помогнат во ублажувањето на овој проблем. Друго ограничување се перформансите на системот во сложени средини. Во реални апликации, возилата може да се преклопуваат или затскриваат едни со други, што го прави предизвик прецизно да се следат поединечни возила. Додека сегашниот систем работи добро за поедноставни сценарија, може да се воведат понапредни техники, како што се алгоритми за следење (на пр., Калман филтри или оптички тек) или модели за длабоко учење (на пр., YOLO, SSD) за справување со посложени средини. Понатаму, системот е првенствено дизајниран за статични камери со фиксни перспективи. Во сценарија каде што камерата може да се движи, background subtraction има помала ефективност. За апликации кои бараат детекција на возила базирани на мобилни или беспилотни летала, други пристапи како оптички проток или следење објекти засновано на длабоко учење би биле неопходни за да се постигнат сигурни резултати.

13. Користена литература

OpenCV документација: https://opencv.org/

FastAPI документација: https://fastapi.tiangolo.com/