Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Документација за проект по предметот Дигитално процесирање на слика

Тема:

Детекција и броење на возила

Ментор:

Проф. д-р Ивица Димитровски

Изработиле:

Михајло Тодоровски 225020

Филип Срезоски 213106

Содржина

I. Вовед……...3

II. Цел на проектот…….……...…..3

III. Имплементација……4

IV. Технологии и библиотеки……...……..5

V. Објаснување на кодот…….7

VI. Заклучок…...……..12

I. Вовед

Во денешно време, со зголемувањето на популацијата и возилата, станува се поизразена потребата за напредни технологии кои ќе помогнат во управувањето и анализата на сообраќајните текови. Во овој контекст, нашиот проект користи техники на дигитална обработка на слики и видео аналитика за детекција и броење на возила во реално време. Системот е развиен со користење на Python и библиотеката OpenCV, што овозможува прецизна и брза обработка на видео податоци.

Главната цел на проектот е да се создаде апликација која нуди следниве функционалности:

- **Детекција на возила:** Апликацијата користи видео за да детектира возила кои поминуваат на одредена локација, овозможувајќи прецизно броење на возила.

Детекцијата на возила има значајни апликации во многу области, вклучително и:

- **Управување со сообраќај:** Оптимизирање на сообраќајните сигнали и планирање на сообраќај за намалување на загушувања.
- **Безбедност и надзор:** Можност за мониторинг на критични локации и подобрување на безбедносните протоколи.
- **Истражување и развој:** Поддршка во развојот на иновативни решенија за интелигентни транспортни системи.

II. Цел на проектот

Проектот за детекција на возила преку видео аналитика има за цел да развие и имплементира софтверска апликација која ќе овозможи ефикасно мониторирање и анализа на сообраќајните текови. Оваа апликација се основа на принципите на компјутерска визија и обработка на видео податоци за да обезбеди точни и релевантни информации кои можат да се користат во различни области како што се управување со сообраќај, планирање на урбани простори, и безбедност. Специфичните цели на проектот вклучуваат:

1. Развој на робустен систем за детекција на возила:

- **Прецизност во детекција**: Системот треба да биде во состојба да точно идентификува возила во различни услови на осветлување и временски услови.
- **Работа во реално време**: Имплементацијата бара да функционира во реално време за да обезбеди актуелни информации за текот на сообраќајот, што е критично за апликации како што се управување со сообраќајните светла и итни реакции.

2. Подобрување на сообраќајното планирање и безбедност:

- **Анализа на податоци**: Собраните податоци ќе се користат за анализа на патни трендови, помагајќи во оптимизација на сообраќајниот тек и планирање на урбаната инфраструктура.
- **Безбедносни аспекти**: Системот ќе помогне во идентификација на потенцијални сообраќајни хазарди и подобрување на мерките за безбедност на патиштата.

3. Поддршка на иновации во интелигентните транспортни системи:

- **Иновативни решенија**: Развојот на овој проект ќе служи како основа за истражување и имплементација на нови технологии во областа на интелигентните транспортни системи.
- **Соработка со академски и индустриски партнери**: Промовирање на соработка помеѓу универзитетите, истражувачките институти и индустријата за развој и интеграција на напредни технологии во практична употреба.

Со овие цели, проектот аспирира не само да го подобри разбирањето и управувањето на сообраќајните текови, туку и да допринесе кон глобалните напори за создавање поодржливи и безбедни урбани средини.

III. Имплементација

Имплементацијата на системот за детекција на возила е фокусирана на создавање ефикасен и надежен софтвер кој користи компјутерска визија за обработка и анализа на видео податоци во реално време. Во овој дел ќе детално опишеме техничките аспекти на развојот на системот, вклучувајќи користените технологии, методи и пристапи:

1. Видео анализа и обработка:

- **Собирање на видео податоци**: Проектот користи видео материјали собрани од CCTV камери насочени кон сообраќајните ленти. Овие видеа се процесираат во реално време за да се идентификуваат возилата.
- **Обработка на слика**: Користиме OpenCV библиотеката за да аплицираме различни техники на обработка на слики како што се филтрирање шум, отстранување на позадината (background subtraction) и детекција на краеви за подобро издвојување на возилата од позадината.

2. Алгоритми за детекција на возила:

- **Примена на MOG2 за отстранување на позадина**: Системот користи MOG2 алгоритам за моделирање и отстранување на позадината за да се обезбеди точно издвојување на движечките објекти (возила).
- **Контурна детекција**: Применуваме методи за контурна детекција за да ги лоцираме и обележиме возилата во видеото. Секое возило е опфатено со правоаголник за подобра визуализација и пребројување.

3. Пребројување и анализа на возила:

- **Линија за детекција**: Дефинираме хоризонтална линија во видеото која служи како точка на детекција. Секое возило кое ја поминува оваа линија се смета за детектирано и се зголемува бројачот на возила.
- **Статистичка анализа**: Податоците за бројот на возила и временските интервали на нивното поминување се користат за статистичка анализа и генерирање на извештаи кои можат да помогнат во планирање и анализа на сообраќајните текови.

4. Тестирање и валидација:

- **Тестирање на системот**: Системот е тестран во различни временски и сообраќајни услови за да се провери неговата надежност и точност. Тестирањето вклучува анализа на перформансите на алгоритмите за детекција и прецизноста на пребројувањето на возила.
- **Оптимизација**: Врз основа на резултатите од тестовите, се прават соодветни оптимизации за подобрување на перформансите и точноста на системот.

Секој од овие елементи придонесува кон целосна и функционална имплементација на проектот, овозможувајќи да се исполни основната цел за точно и ефикасно детектирање и анализа на сообраќајните текови.

IV. Технологии и библиотеки

Проектот за детекција на возила користи различни технологии и библиотеки за обработка и анализа на видео податоци. Изборот на соодветни алатки е клучен за успешноста на проектот. Во овој дел ќе опишеме детално кои технологии и библиотеки се користени во проектот и зошто тие се избрани:

1. Python:

- **Зошто Python?**: Python е еден од најпопуларните програмски јазици за обработка на податоци и машинско учење поради својата голема заедница и широк опсег на библиотеки. Тој овозможува лесна интеграција на различни алатки и библиотеки, што го прави идеален за проекти кои бараат сложена обработка на податоци и видео аналитика.

2. OpenCV:

- **Основни функционалности**: OpenCV (Open Source Computer Vision Library) е водечка библиотека наменета за компјутерска визија. Таа нуди богат сет на функции за обработка на слики и видео, вклучувајќи алгоритми за отстранување на позадина, детекција на објекти, и следење на движења.
- **Примена во проектот**: Во рамките на овој проект, OpenCV се користи за да се обработат видео снимки во реално време, идентификувајќи и пребројувајќи возила. Алгоритмите за отстранување на позадина како MOG2 и техники за контурна детекција се клучни за точноста на детекцијата.

3. NumPy:

- **Основни функционалности**: NumPy е основна библиотека за научни пресметки во Python. Таа овозможува поддршка за големи мултидимензионални низи и матрици, заедно со збир од математички функции за работа со овие низи.
- **Примена во проектот**: NumPy се користи за ефикасна манипулација и анализа на податоци извлечени од видео снимките, што овозможува брза обработка и математички пресметки потребни за детекција на возила.

4. Matplotlib:

- **Основни функционалности**: Matplotlib е библиотека за цртање графикони во Python. Таа е многу корисна за визуелизација на податоци, што е суштински дел од анализата на податоците.
- **Примена во проектот**: Во контекстот на овој проект, Matplotlib се користи за да се генерираат визуелизации на сообраќајните анализи, како што се графикони на пребројување на возила во различни временски интервали.

Алгоритми за одземање на позадина:

- **MOG2 (Mixture of Gaussians 2):** Овој алгоритам моделира секој пиксел во сликата како мешавина од Гаусови распределби. Тоа овозможува динамично прилагодување на моделот на позадината, што е корисно при промени во осветлувањето или движење на позадината (на пр. дрвја кои се движат на ветрот).

Морфолошки операции:

- **Затворање и отворање:** Овие операции се комбинации од ерозија и дилатација. Затворањето помага во пополнување на мали дупки во објектите, додека отворањето отстранува мали објекти или шумови од сликата.

Детекција на контури:

- **cv2.findContours:** Оваа функција пронаоѓа контури во бинарна слика. Контурите се корисни за анализа на формата и големината на објектите во сликата.

Пресметка на центарот на објектите:

- Функцијата get_center пресметува центар на правоаголникот кој го обградува детектираниот објект, што е важно за следење и броење на објектите.

Комбинацијата на овие технологии и библиотеки создава моќна платформа која овозможува ефикасна реализација на целите на проектот, пружајќи точни и корисни инсајти за сообраќајната динамика.

V. Објаснување на кодот

Кодот за детекција на возила е поделен во неколку клучни сегменти кои заедно обезбедуваат целосна функционалност за обработка на видео податоци и анализа на возила. Секој сегмент е детално разгледан подолу.

```python
import cv2

import numpy as np
```

Ги вклучуваме библиотеките **OpenCV** и **NumPy**, потребни за обработка на видеото и работа со нумерички низи.

```python
video_path = 'video.mp4'
cap = cv2.VideoCapture(video_path)
```

Го дефинираме патот до видеото и креираме објект VideoCapture за читање на фрејмови од видеото.

```python
bg_subtractor = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=50)
```

Креираме објект за одземање на позадината користејќи **MOG2** алгоритам со зададени параметри: историја од 500 фрејмови и праг на варијација од 50.

```python
line_position = 450
```

```
vehicle_count = 0

min_contour_width = 40

min_contour_height = 40

offset = 6
```

- line_position: Поставуваме позиција на линијата за броење на 450 пиксели по вертикала.
- vehicle_count: Иницијализираме броење на возила на нула.
- min_contour_width и min_contour_height: Дефинираме минимални димензии на контура за да се смета како возило.
- offset: Дефинираме офсет за детектирање на возила во близина на линијата за броење.

```python
def get_center(x, y, w, h):
    x1 = int(w / 2)
    y1 = int(h / 2)
    cx = x + x1
    cy = y + y1
    return cx, cy
```

Дефинираме функција get_center за пресметка на центарот на детектираниот објект базирано на координатите и димензиите на правоаголникот.

```python
detects = []
```

Иницијализираме листа detects за складирање на координатите на детектираните центри на објектите.

```python
while True:
    ret, frame = cap.read()
    if not ret:
        break
```

Започнуваме бесконечна петља за читање на фрејмови од видеото. Ако нема повеќе фрејмови (ret е False), излегуваме од петљата.

```python
   frame = cv2.resize(frame, (800, 600))
```

Променуваме големина на фрејмот на **800x600** пиксели за конзистентна обработка.

```python
   fg_mask = bg_subtractor.apply(frame)
```

Применуваме одземање на позадината за да го добиеме **предниот план** (движечките објекти) во фрејмот.

```python
   _, fg_mask = cv2.threshold(fg_mask, 244, 255, cv2.THRESH_BINARY)
```

Применуваме праговирање за да ја претвориме маската во **бинарна слика**, каде што пикселите со вредност над 244 стануваат бели (255), а останатите црни (0).

```python
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_CLOSE, kernel)
    fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)
```

- Креираме **структурен елемент** со големина 5x5 за морфолошки операции.
- Применуваме **морфолошко затворање** за пополнување на мали дупки во предниот план.
- Применуваме **морфолошко отворање** за отстранување на шумови и мали објекти.

```
Python

contours, _ = cv2.findContours(fg_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Наоѓаме контури во бинарната слика кои претставуваат потенцијални возила.

```
Python

cv2.line(frame, (0, line_position), (frame.shape[1], line_position), (255, 0,
0), 2)
```

Цртаме хоризонтална линија на фрејмот која ќе се користи за броење на возилата кога ќе ја преминат.

```
Python
for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)
        if w >= min_contour_width and h >= min_contour_height:
            center = get_center(x, y, w, h)
            detects.append(center)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

- За секоја контура која ги задоволува минималните димензии:
    - Пресметуваме обграден правоаголник.
    - Го пресметуваме центарот на правоаголникот.
    - Го додаваме центарот во листата detects.
    - Цртаме правоаголник околу објектот на фрејмот.

```
Python
   for (x, y) in detects:
```

```
        if (line_position - offset) < y < (line_position + offset):
            vehicle_count += 1
            detects.remove((x, y))
            print(f"Vehicle Count: {vehicle_count}")
```

- За секој детектиран центар:
    - Проверуваме дали центарот ја преминал линијата за броење со одреден офсет.
    - Ако да, го зголемуваме бројачот за возила и го отстрануваме центарот од листата detects за да избегнеме двојно броење.
    - Го печатиме тековниот број на возила.

```
Python
    cv2.putText(frame, f"Vehicle Count: {vehicle_count}", (10, 50),
                cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 3)
```

Го прикажуваме тековниот број на возила на фрејмот, на позиција (10, 50) со црвен текст.

```
Python
    cv2.imshow('Video', frame)
     cv2.imshow('Foreground Mask', fg_mask)
```

Прикажуваме два прозорци:

- 'Video': со оригиналниот фрејм и надредени графики (линија за броење, правоаголници, бројач).
- 'Foreground Mask': со бинарната маска на предниот план.

```
Python
    key = cv2.waitKey(30)
     if key == 27:
         break
```

- Чекаме 30 милисекунди за притисок на копче.

- Ако е притиснато **ESC** (код 27), излегуваме од петљата и го прекинуваме процесот.

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

- Го ослободуваме објектот VideoCapture и ги затвораме сите отворени OpenCV прозорци за чисто завршување на програмата.

VI. Заклучок

Во оваа семинарска работа, се осврнавме на имплементацијата на систем за детекција и броење на возила во видео запис, користејќи ги библиотеките OpenCV и NumPy. Преку анализата на кодот, идентификувавме клучни компоненти и техники кои овозможуваат прецизна и ефикасна обработка на видео податоци.

Клучни аспекти на системот:

1. **Одземање на позадината со MOG2 алгоритам**
   - Динамична адаптација: MOG2 овозможува прилагодување на позадината во реално време, што е критично за видеа со променливи услови на осветлување.
   - Ефикасност: Со употреба на историја од 500 фрејмови и праг на варијација од 50, системот успешно ги изолира подвижните објекти.
2. **Морфолошка обработка на сликата**
   - Затворање и отворање: Примената на овие операции помага во отстранување на шумот и пополнување на дупките во бинарната слика.
   - Подобрување на контурите: Ова резултира со почисти контури кои се полесни за детекција и следење.
3. **Детекција на контури и филтрирање**
   - Минимални димензии на објекти: Со поставување на праг за минимална ширина и висина, се избегнуваат лажни позитиви од мали и незначајни објекти.
   - Пресметка на центарот на објектот: Ова е клучно за следење на објектите и утврдување дали ја преминале линијата за броење.
4. **Броење на возила преку линија за детекција**
   - Линија со офсет: Поставувањето на линијата со одреден офсет осигурува дека возилото навистина ја преминало линијата, а не само се приближило до неа.
   - Избегнување на двојно броење: Отстранувањето на детектираниот центар од листата по броењето спречува повторно броење на истото возило.
5. **Визуелизација и интеракција со корисникот**

- Графички приказ: Прикажување на линијата за броење, обградувачките правоаголници и бројот на возила директно на фрејмот.
- Реално време: Можност за мониторинг во реално време преку интерфејсот на OpenCV.

Предности на системот:

- Едноставност и лесно разбирање: Кодот е структуриран и коментиран, што го прави погоден за образовни цели и брзи прототипови.
- Прилагодливост: Параметрите како што се позицијата на линијата, минималните димензии и офсетот лесно можат да се модифицираат за различни сценарија.
- Ефикасност во обработката: Употребата на основни техники за обработка на слика овозможува брза обработка дури и на послаби системи.

Потенцијални подобрувања и идни насоки:

1. **Интеграција на напредни алгоритми за машинско учење**
   - Длабоко учење: Користење на конволуциски невронски мрежи за подобра класификација на објектите и намалување на лажните позитиви.
   - Препознавање на типови на возила: Проширување на системот за да класифицира различни категории на возила (автобуси, камиони, мотоцикли).
2. **Оптимизација за обработка во реално време**
   - Паралелна обработка: Искористување на GPU за забрзување на морфолошките операции и детекцијата на контури.
   - Мултипроцесирање: Поделба на задачите на повеќе процеси или нишки за подобра искористеност на ресурсите.
3. **Адаптивни параметри и автоматизација**
   - Автоматско прилагодување на праговите: Користење на алгоритми кои динамично ги прилагодуваат параметрите во зависност од условите на околината.
   - Учење од податоци: Системот да собира податоци и да се подобрува со текот на времето преку машинско учење.
4. **Интеграција со други системи**
   - IoT и сензори: Комбинирање на видео анализата со податоци од други сензори за поцелосна слика на сообраќајот.
   - Информациони системи за сообраќај: Поврзување со градски системи за управување на сообраќајот за подобрување на протокот и безбедноста.

Заклучни размислувања:

Оваа имплементација претставува солидна основа за разбирање на концептите на компјутерскиот вид и неговата примена во реални сценарија. Преку комбинација на основни техники и алгоритми, успеавме да создадеме функционален систем кој може да се користи како почетна точка за понапредни проекти.

Идни можности:

- Образование и истражување: Системот може да се користи во образовни установи за демонстрација на техники за обработка на слика.
- Практични апликации: Со понатамошен развој, системот може да се интегрира во решенија за мониторинг на сообраќајот, безбедност и аналитика.
- Заедница и соработка: Отвореноста на кодот овозможува придонес од заедницата, што може да доведе до побрз развој и иновации.

Заклучок:

Со анализата и имплементацијата на овој систем, демонстриравме како релативно едноставни техники можат да решат сложени проблеми. Оваа работа не само што придонесува кон нашето знаење во областа на компјутерскиот вид, туку и отвора врати за понатамошни истражувања и практични примени кои можат да имаат позитивно влијание врз општеството.

Дополнително, оваа работа придонесува кон нашето колективно знаење во областа, нудејќи практичен пример за тоа како може да се имплементира систем за детекција и броење на возила со користење на достапни технологии и библиотеки. Ова може да послужи како основа за понатамошни истражувања и развој, особено во контекст на интелигентните транспортни системи, урбаното планирање и управувањето со сообраќајот. Со оглед на тоа што сообраќајните проблеми стануваат се поизразени во модерните градови, ваквите системи можат да имаат значително позитивно влијание врз општеството, овозможувајќи подобра ефикасност, намалување на застојот и подобрување на безбедноста на патиштата.

Исто така, нашата имплементација отвора врати за интеграција на напредни техники за машинско учење и вештачка интелигенција, кои би можеле дополнително да ја подобрат точноста и функционалноста на системот. На пример, со додавање на модели за длабоко учење, системот може да се оспособи за класификација на различни типови на возила, предвидување на сообраќајни текови или дури и детекција на инциденти во реално време. Ова би имало директна корист за општеството, преку подобрување на услугите во транспортниот сектор и придонес кон создавање на попаметни и поодржливи градови.

Понатаму, овој проект има потенцијал да биде адаптиран за различни средини и услови, што го прави флексибилен и применлив во глобален контекст. Може да се користи во различни земји и градови, прилагодувајќи се на специфичните потреби и инфраструктура. Со оглед на глобалните предизвици поврзани со сообраќајот, загадувањето и урбанизацијата, ваквите решенија се од големо значење за подобрување на квалитетот на животот.

Нашиот пристап исто така ја нагласува важноста на отворените технологии и софтвер, кои овозможуваат пристапност и можност за соработка меѓу истражувачите и практичарите. Со споделување на нашите наоди и методологии, поттикнуваме заедничко учење и иновации, што може да доведе до побрз напредок во областа. Ова исто така може да

поттикне создавање на стандарди и најдобри практики кои ќе бидат корисни за целата заедница.

Во целина, овој проект ја илустрира важноста на иновативните решенија базирани на технологија во справувањето со современите предизвици. Тој потврдува дека со креативност, посветеност и соработка, можеме да развиеме системи кои не само што ги задоволуваат техничките барања, туку и придонесуваат за општото добро. Ова претставува значаен чекор напред во нашите напори да изградиме поодржливо и поврзано општество.

Референци:

1. **https://docs.opencv.org/4.x/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html**
2. **https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html**
3. **https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/**
4. **https://www.mathworks.com/help/supportpkg/opencv/ref/vehicledetectionexample.html**