|  |  |  |
| --- | --- | --- |
| **images** | **УНИВЕРЗИТЕТ “Св. КИРИЛ И МЕТОДИЈ” - СКОПЈЕ**  **ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ** | **index** |

- **ПРОЕКТНА РАБОТА** -

по предметот

**МАШИНСКИ ВИД**

**Тема**

**АВТОНОМНО ВОЗИЛО**

|  |  |  |
| --- | --- | --- |
| Ментор: | Изработил: | |
| Проф. д-р Зоран Ивановски | | Васил Трендафилов, индекс бр. 139/2018  e-mail: vasetrendafilov@gmail.com |

*Скопје, јули 2022*

**Содржина**

[Апстракт 2](#_Toc108379268)

[Вовед 3](#_Toc108379272)

[1 Препознавање летна на патот 3](#_Toc108379273)

[1.1 Калибрација на сликата 5](#_Toc108379274)

[1.2 Тест на испрекината лента 6](#_Toc108379275)

[1.3 Процесирање на лентите 6](#_Toc108379276)

[1.4 Работи што треба да се подобрат 7](#_Toc108379277)

[1.5 Работи што треба да се додадат 7](#_Toc108379278)

[2 Детекција на возила 7](#_Toc108379279)

[3 Заклучок 8](#_Toc108379280)

[4 Референци 9](#_Toc108379281)

# Апстракт

*Автономните возила сега се неопходност во овој свет, што го прави апсолутно важно да се има звучен алгоритамски систем кој ќе врши откривање на различни карактеристики на патот, како што се маркери за лента, откривање возила итн. Во овој проект, лентите ќе бидат откриени со користење на canny и hough lines и алгоритам за филтрирање на трите ленти. Ќе ја имплементираме и невронската мрежа yolov5 за откривање возила на патот. Овој пајплајн во реално време ќе се имплементира на duckiebot за тестирање на перформансите на алгоритмот.*

# Клучни зборови: компјутерски вид, детекција на лента, автономно возење, yolov5

**Abstract**

# *Self driving cars are now a necessity in this world, which makes it absolutely important to have a sound algorithmic system that would perform detection of various features on the road, such as lane markers, detecting vehicles etc. In this project, lanes will be detected using canny and houg lines and an algorithm to filter out the three lanes. We will also implement the neural network yolov5 for detecting vehicles on the road. This real-time pipeline will be implemented on the duckiebot to test the performance of the algorithm.*

# Keywords: computer vision, lane detection, autonomous driving, yolov5

# Вовед

Зголемувањето на безбедноста, намалувањето на сообраќајните несреќи и зголемувањето на удобноста и искуството во возењето се главните мотиви за опремување на модерните автомобили со напредни системи за помош при возење (ADAS - Advanced Driving Assistance Systems). Во изминатите неколку децении, големите производители на автомобили воведоа многу софистицирани функции како што се предупредување за напуштање лента, помош за задржување на лента, електронска контрола на стабилноста, итн. Идните функции на ADAS како избегнување судир, автоматско возење на автопат (автопилот), автоматско паркирање и кооперативно маневрирање бараат се побрзо и посигурно откривање на границите на патиштата, што е меѓу најкомплексните и најпредизвикувачките задачи.

Функционалноста за откривање на границите на патот бара локализација на патот, одредување на релативната позиција помеѓу возилото и патот и анализа на насоката на движење на возилото. Ќе разгледаме алгоритам за препознавање на лентите и детекција на автомобил со невронската мрежа yolov5 [2].

# Препознавање летна на патот

Техниките за компјутерска визија се главните алатки кои обезбедуваат способност за откривање, идентификација и следење на коловозните ленти. Откривањето на ленти главно се состои од пронаоѓање на специфични карактеристики како што се ознаките на лентата (обоени површини на патиштата). Сепак, постојат некои ситуации, каде може да го попречат откривањето на лентата. Како пример, постоењето на други автомобили на истата лента што ги крие, целосно или делумно, друг пример е постоењето на расфрлани региони во сенки предизвикани од ѕидови на автопат, згради, дрвја, итн.

Во овој проект, ќе разгледаме пристап заснован на визија способен да постигне перформанси во реално време при откривање и следење на лентите. Алгоритамот за препознавање на лентата се наоѓа во lane\_datection.ipynb [1] каде што алгоритмите се тестираат на слики направени од возилото. Вистинската имплементација на алгоритмот е во папката duckiebot [1] кој е докер рос контејнер каде што комуницира со останатите компоненти на автомобилот и ги контролира тркалата. Едно од клучните проблеми е времето на извршување, но успеав да стигам до околу 30 мили секунди по слика што е доволно за да се контролира возилото. Развиен е сликовен пајплајн за наоѓање коловозни ленти кој е опишан со следните чекори:

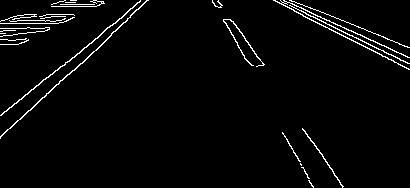


Слика од роботот и мапата











Како дополнителни можности во алгоритамот имаме преспективна проекција на патот со поглед одозгора за полесни и прецизни калкулации за каде да се постави возилото на патот. Четирите точки се одбираат како пресек на лентите со рамката на сликата и може да се одберат трите опции да го гледа левиот или десниот дел од патот или целиот заедно. Исто така имаме и потенцијални подобрувања за пронајдување на лентите како филтрирање на сликата со фиксен или адаптивен праг на сивата слика. Фиксниот е многу побрз од адаптивниот, но има проблеми со промена на светлината, доволно е ламбата во собата да се исклучи и да неможе да ги филтрира лентите.



Фиксен филтер каде ги пропушта сите пиксели над 160



Поглед одозгора

Адаптивен филтер каде големината на блокот е 55 и С е -6

## Калибрација на сликата

За калибрација на сликата се користи методот на шаховска табла од OpenCV [3]. Сите слики земени со шаховска табла се наоѓаат во папката calibration [1]. Направена е едноставна тетратка за калибрација во јупитер за да се најдат параметрите. Се користат специјални знамиња и критериуми за леќата на рибното око на возилото duckiebot. Последното поле во тетратката се користи за прикажување на разликите помеѓу необработената и калибраната слика. Средната квадратна грешка при калибрација е 0.42 при добра репрезентација на шаховската табла од сите гледни точки со што сликата испадна прилично добро.



Калибрирана слика

Необработена слика

## Тест на испрекината лента

Една од најтешките работи е да се провери која од групираните ленти е испрекината затоа што има многу шум, а понекогаш дури и не може да ги препознае испрекинатите сегменти на лентата со hough lines [5]. Функцијата мине во опсег од почеток кон крај на висината на сликата со одреден чекор и проверува на секоја y позиција колку линии се пресекуваат со таа права. Ако се сече со најмалку една линија и го повторува тоа над прагот за одредување, се брои како премин. На крајот дава одговор ако бројот на транзиции е поголем или еднаков од наведената транзиција во аргументот. Најдобрите параметри што досега функционираат се чекорот е 5, прагот е 2 и транзицијата е 4.

## Процесирање на лентите

Најтешко е да се пронајдат вистинските ленти со целиот шум на патот и предвидената шаховска табла како почетна плочка. Како главен влез во оваа функција се линиите што се генерираат со hough lines [5], првата работа е да се трансформираат од пар точки во наклон и офсет и да се филтрираат оние што не се во опсегот на наклон од 0,2 до 3. Исто така, се зачувува растојанието x каде се сече со правата на половина од сликата за да се подредат линиите. Со функцијата grouper [1] се групираат линиите во лента ако растојанието помеѓу линиите е помал од прагот, засега работи прилично добро со праг од 30. Одлучувањето на лентите е поделен на два дела:

Додека баферот на минатите потенцијални ленти не е пополнет над аргументот минато, се бараат трите ленти секој фрејм и се зачувуваат во баферот. Прво се наоѓа испрекината лента каде се отстранува од групираните ленти и се наоѓа растојанието x на пресекот со половина од сликата. Потоа се одзема со другите групирани ленти и се добива должината меѓу нив. За левата и десната лента се одбираaт најблиските ленти до испрекираната. Ако го задоволат последниот услов се додаваат во баферот.

Откако ќе се наполни баферот, се користи за да имаме побрзи и посигурни предвидувања. Прво се добиваат трите минати ленти со просек на минатите ленти во баферот и се наоѓаат најблиските ленти до нив. Новите потенцијални ленти се потврдуваат ако разликата во пресекот е помала од 40. Следно имаме многу услови:

* Ако нема мината и потенцијална испрекината лента, повторно се движиме низ сите ленти за да се пронајде,
* Ако нема потенцијална лева лента и има испрекината лента, тогаш се повторува истиот алгоритам за да се најде левата лента и дополнително го имаме растојанието до средната лента што се користи за прецизно да ја одреди лентата,
* Ако нема потенцијална десна лента и има испрекината лента, тогаш се повторувам истиот алгоритам за да се најде десната лента и дополнително го имаме растојанието до средната лента што се користи за прецизно да ја одреди лентата.

Следни се последните проверки за да се избере која лента да се предвиди. Ако сите услови се исполнети баферот се полни со новите ленти и се отфрлаат најстарите:

* Ако средната линија не е пронајдена, се прави основна проверка за левата и десната лента и се ажурира растојанието со празно расојание,
* Инаку, се проверува левата и десната лента од средината и се ажурира растојанието до неа,
* Последната проверка е да се врати потенцијалната лента ако постои.

## Работи што треба да се подобрат

* Груперот да направи фина разлика помеѓу лентите со наоѓање на левата и десната страна на лентата и да ги комбинира,
* Функција која проверува дали лентата е полна за да се отстрани шумот,
* Предвидувања за лентите што се блокирани од други возила за да ги подобрите горенаведените проверки,
* Рангирана просечна лента или предвидување каде можат да биде зависно од поместувањето на возилото,
* Подетални финални проверки.

## Работи што треба да се додадат

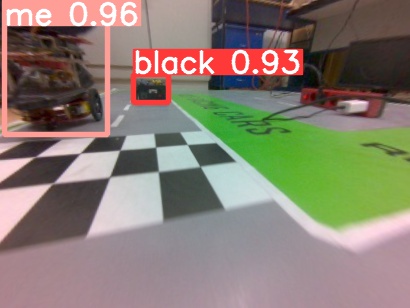
* Нова функција за филтрирање на десна или лева кривина,
* Предвидување на ленти иако не постојат на сликата,
* Користење на проекцијата одозгора за да се пресмета прецизно движење на возилото.

# Детекција на возила

Невронската мрежа што се користи за детекција на возилата е yolov5 [2]. Дизајнирана да биде погодна за корисниците бидејќи е имплементирана во pytorch и со помош на google colab е лесно да се прилагоди на мојот случај. Чекорите за тренирање на мрежата се:

1. Анотирање на сликите со помош на онлајн алатката Cvat [6],
2. Со roboflow се прикачуваат сликите, се користи агментација (промена на осветлувањето, заматување на сликата и додавање на шум) за зголемување на податоците и креира база на податоци за тренирање и валидација [7],
3. Користејќи ја тетратката на google colab [8] за фино прилагодување на мрежата на мојата база на податоци користејќи ги нивните достапни бесплатни графички,
4. Превземување на мрежата на мојот уред и тестирање на нови слики.

Пробани многу конфигурации за yolov5 бидејќи доаѓа со различни мрежни архитектури. Најприфатлива конфигурација е yolo5n, која е најмалиот и најбрзиот модел од купот и за мојата употреба, работеше одлично. Користев 150 епохи и група од 15 слики. Тренираните модели може да се најдат во yolov5\runs\train [1]. Најневеројатните резултати беа од тест сликите, првпат ја пробав мрежата на возилата одпозади, резултатите можете да ги видите во yolov5\runs\detect [1].



Слика од множеството за тестирање

Слика на возилата одпозади

# Заклучок

Во овој проект, развиена е брза и сигурна техника за откривање и следење на лентите на патот. Користи пајплајн од добро познати алгоритми како откривање на рабовите Canny и трансформација на Hough. Покрај тоа, пајплајнот користи сеопфатна техника за одлучување на трите моторни ленти. За предложената техника потребни се само необработени слики од една камера поставена на предниот дел од возилото. Резултатите од валидацијата покажуваат прилично прецизно и робусно откривање, освен во едно сценарио каде што постои шаковската табла како старт тајл. Измереното време на извршување со помош на процесорот на јетсонот докажа дека е многу погоден за откривање лента во реално време без многу трошоци. Затоа, предложената техника е добро прилагодена за употреба во напредни системи за помош при возење (ADAS) или самоуправувачки автомобили. Сите пречки на лентите може да се елиминираат со помош на невронската мрежа yolov5 што ја нагодивме за нашиот пример. Кога ќе дојде прашањето за препознавање на лева или десна кривина, кружен тек или раскрсница и да не заборавиме сите знаци веќе е многу тешко да се направи систем кој ќе биде посигурен од самиот човек што го вози. Во идеални услови како што е мапата на кое е направено тестирањето имаме добри резултати и е многу добар концепт за развивање на автономно возило, но за да работи во реалниот свет има уште многу работа и многу услови и параметри кои треба да се нагодат.

# Референци

1. <https://github.com/vasetrendafilov/self-driving-car>
2. <https://github.com/ultralytics/yolov5>
3. <https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html>
4. <https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html>
5. <https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html>
6. <https://cvat.org/>
7. <https://universe.roboflow.com/feit-iv6em/jetbot-rwmjo>
8. <https://colab.research.google.com/drive/14_b5AdCSUyWwgaDhfHkinvDp_pjaqagx?usp=sharing>