МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп‘ютерних наук

Кваліфікаційна робота магістра

ІНФОРМАЦІЙНА СИСТЕМА РОЗПІЗНАВАННЯ ВЕЛОСИПЕДНОГО ТРАНСПОРТУ

Здобувач освіти гр. ІН.м – 13 Кирило БАЛАЦЕНКО

Науковий керівник,
ст. викл., к.т.н. Артем КОРОБОВ

В. о. завідувача кафедри
доцент, к.т.н. Ігор ШЕЛЕХОВ

Суми 2022

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«122 - Комп'ютерні науки»*

Затверджую:
В.о. зав.кафедри _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Балаценко Кирилу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Інформаційна система розпізнавання велосипедного транспорту*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі здобувачем вищої закінченого роботи_____

3. Вхідні дані до роботи _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Огляд технологій, що застосовуються для розпізнавання об'єктів, машинного навчання, комп'ютерного зору; 2) Постановка задачі та формування завдань дослідження; 3) Опис алгоритму розпізнавання велосипедного транспорту; 4) Розробка додатку; 5) Аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата

		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Огляд технологій, що застосовуються для розпізнавання об’єктів, машинного навчання, комп’ютерного зору		
2.	Постановка задачі та формування завдань дослідження.		
3.	Опис алгоритму розпізнавання велосипедного транспорту		
4.	Розробка додатку для розпізнавання велосипедного транспорту		
5.	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 32 стор., 14 рис., 1 додаток, 10 літературних джерел.

Об’єкт дослідження — Інформаційна система розпізнавання велосипедного транспорту.

Мета роботи — створення інформаційної системи попередження дорожньо-транспортних пригод (або зіткнення) велосипедного транспорту шляхом використання технологій комп’ютерного зору та машинного навчання

Результати — проведений аналіз літератури, методів та інструментів, які дозволяють розпізнавати та відстежувати певні об’єкти на фото- та відеоматеріалах за допомогою технологій комп’ютерного зору. Після ознайомлення з існуючими рішеннями було розроблено алгоритм розпізнавання велосипедного транспорту та його реалізація у вигляді застосунку. Дана розробка дозволяє розпізнавати велосипедний транспорт для різних задач (запобігання аварійним ситуаціям, відстеження учасників змагань). Застосунок був реалізований за допомогою мови програмування Python.

ІНФОРМАЦІЙНА СИСТЕМА РОЗПІЗНАВАННЯ ВЕЛОСИПЕДНОГО ТРАНСПОРТУ, INFORMATION SYSTEM FOR BICYCLE RECOGNITION, OPENCV, PYTHON, NUMPY

ЗМІСТ

ВСТУП

Сьогодні неможливо уявити світ без використання мультимедійної апаратури, особливо пов’язаної з відео контентом. Платформи для трансляції, системи відеоспостереження, кінематограф – все це об’єднує те, що кінцевим продуктом є саме відеоряд. Проте, в кожному з цих випадків відрізняється цільове призначення кінцевого продукту. Трансляції мають широкий спектр кінцевих цілей – користувач має змогу переглянути наживо або в повторі змагання, цікаве шоу, важливі суспільні теми тощо. Системи відеоспостереження дозволяють слідкувати за порядком у певних місцевостях та допомагають шукати тих, хто його порушує. Кінематограф забезпечує кінцевим користувачам естетичне задоволення, інше життя за допомогою сюжетної лінії, довідку про історичні факти.

В більшості випадків, відео контент неможливо уявити без використання технологій машинного навчання, штучного інтелекту. Якби ми зараз розмовляли про це 20-30 років тому, в голові ми би уявляли SkyNet з саги «Термінатор», але зараз це буденність. Візьмемо, наприклад, смартфон компанії Apple iPhone. Даний смартфон має технологію FaceID, що повністю аналізує ваше обличчя та у подальшому дозволяє розблоковувати пристрій, просто поглянувши на свій смартфон. На відміну від технології розблокування за обличчям в аналогічних android-смартфонах, FaceID помітить підміну за допомогою фото або відеоряду, адже ще на етапі створення так званого «відбитку» аналізуються тисячі точок інтересів вашого обличчя і цей процес відбувається настільки точно, що різницю між справжнім обличчям та світлиною з ним смартфон помітить лише за декілька секунд. Окрім цього, iPhone має голосового асистента Siri, який по сьогоднішній день вважається одним з найкращих через свою багатогранність. Siri може знайти для вас будь що, бути хорошим співрозмовником, вирішити прості задачі, а головне – постійно вдосконалюється за рахунок самонавчання.

Зупинимось на технології FaceID. Чому саме на ній? По суті, тут мають місце технології запам’ятовування та подальшого розпізнавання певного

об’єкта, в даному випадку обличчя користувача. Системи відеоспостереження у великих містах розвинених країн, торгових центрах, мають супутні системи розпізнавання. У випадку з торговими центрами, вони можуть використовуватися для маркетингових цілей – підрахунок кількості відвідувачів, настрій клієнтів під час програвання тої чи іншої реклами, на вивіску якого з магазинів більше відвідувачів звертають увагу тощо. Якщо ми кажемо про великі міста, тут більш специфічне призначення для технологій розпізнавання. За допомогою систем відеоспостереження у сукупності з технологіями розпізнавання, можливе визначення потенційно небезпечних громадян, що пересуваються вулицею – в даному випадку сигналізується про підозрілість перехожого, що призводить до його подальшого затримання правоохоронними органами. Також такі системи дозволяють миттєво реагувати на надзвичайні ситуації, що можуть відбуватися на вулиці – наприклад, дорожньо-транспортні пригоди або терористичні акти. Для прикладу, аеропорт Бен-Гуріон в Ізраїлі. Як відомо, весь аеропорт містить в собі представників правоохоронних органів, військових та найманих охоронців в уніформі охорони аеропорту. Проте також відомим є той факт, що ще за 50 км до нього починається стеження за кожним, хто планує туди потрапити. Тобто, які б плани не були у того чи іншого потенційного відвідувача аеропорту, в будь-який момент його оперативно зможуть перехопити правоохоронці та запобігти можливому порушенню закону (терористичний акт, контрабанда тощо). І все це не завдяки стеженню конкретними людьми, а і також завдяки технологіям розпізнавання.

Чи є широкодоступними технології розпізнавання, штучного інтелекту, машинного навчання? Наразі існує безліч рішень з відкритим вихідним кодом, які при правильному підході задовольнять потреби розробників та їх замовників. Найпопулярнішими такими рішеннями є OpenCV, TensorFlow, CUDA. Ці рішення втілюються за допомогою певних мов програмування і якщо розглядати варіанти для ознайомлення або навіть реального використання, то найпростішою

та найпоширенішою для цього мовою програмування є Python, яка займає першу позицію в рейтингу TIOBE.

Не дивлячись на розгляд у даному вступі технологій розпізнавання для маркетингу та цивільної безпеки, темою даної роботи є інформаційна система розпізнавання велосипедного транспорту. Дана система буде вузько направленою, проте дозволить вирішувати певні задачі, а саме: запобігання дорожньо-транспортних пригод у певних ділянках міста, відслідковування учасників змагань на велотрекінгу, протидія крадіжці велосипедного транспорту тощо.

Для вирішення даної задачі буде розглянуто відомі інструменти комп’ютерного зору та машинного навчання, проведено аналіз мов програмування та обрано конкретну мову, за допомогою якою буде досягнуто потрібних цілей нашої роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

На сьогоднішній день більшість представників української наукової сфери стверджують, що без технологій розпізнавання не може обійтися майже жодна система відеоспостереження або система відстеження проведення змагань з певних видів спорту.

Наприклад, існує система відео асистента рефері (video assistant referee, VAR) у футбольних змаганнях, яка дозволяє арбітру переглянути спірний момент та прийняти рішення, від якого залежатиме подальший хід матчу. Даний процес не є автоматизованим, через що 95% футбольних фанатів вважають цей процес необ’єктивним.

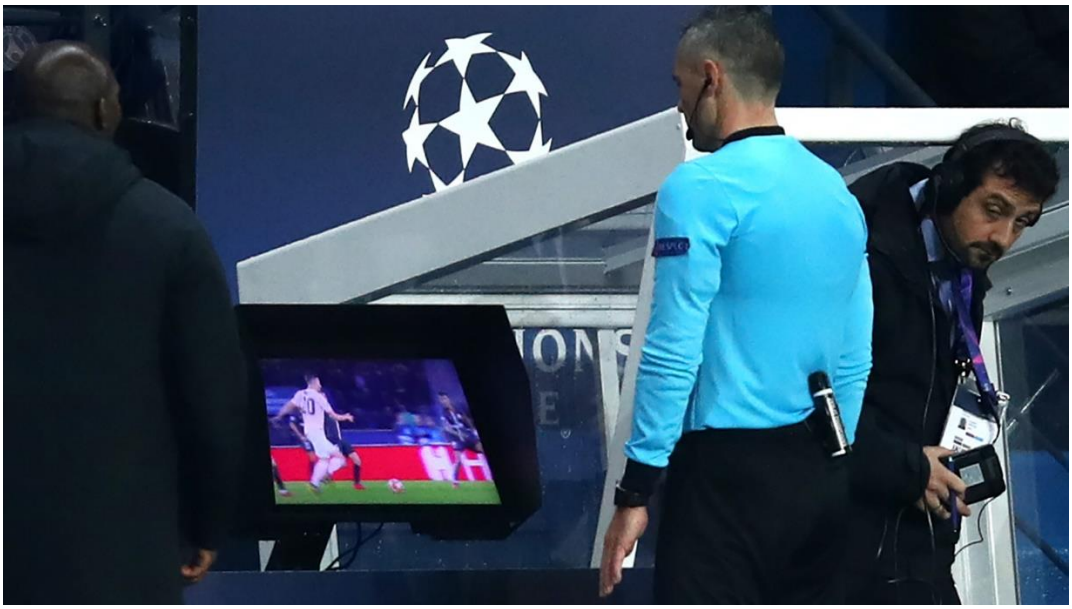

Рисунок 1.1 – футбольний рефері під час роботи з системою VAR

Як технології розпізнавання можуть вплинути на цю систему? По-перше, таким чином можливе нівелювання людського фактору, який може зіграти злий жарт, адже завжди при використанні системи VAR на рефері покладається обов’язок прийняти складне рішення, з яким в ідеалі мають погодитися обидві сторони цього матчу. Саме погодитися, бо одній стороні вигідно, щоб іншій не

зарахували «Гол» або «Штрафний», але правильне рішення у подальшому не викличе чимало петицій або апеляцій.

По-друге, витрачатиметься набагато менше часу, адже саме технології комп’ютерного зору та машинного навчання зможуть приймати більш точні рішення менше ніж за хвилину, у порівнянні з рефері, через перегляд якого може бути втрачено цінні 3-5, а то і більше хвилин під час прийняття рішення.

Не лише у футболі використовується дана система і не лише у ньому даний процес не автоматизовано, хоча є всі технічні можливості для цього.

Якщо до уваги брати велосипедний спорт, то в цьому випадку розпізнавання та відслідковування важливе за таких причин:

1. Нещасні випадки. Нерідко під час змагань на велотрекінгових аренах існує загроза зіткнення двох велосипедистів. Це відбувається через високі швидкості самих спортсменів, також через їх недостатній досвід. На такій високій швидкості складно відслідкувати випадки, коли станеться зіткнення або падіння, яке призведе до паралізації інших учасників та унеможливить подальший хід змагань.
2. Визначення переможця. Як вже було зазначено, такі змагання зазвичай проводяться на високих швидкостях, а отже людське око, людський фактор може невірно визначити переможців перегонів.
3. Статистичні дані. Наприклад, на змаганнях потрібно визначити рекордсмена за швидкістю проходження перегонів, за найвищою швидкістю під час кола, за маневреністю. Людський фактор також може цього не відслідкувати.

Рисунок 1.2 – змагання на велотрекінговій доріжці

Технології комп’ютерного зору здатні вирішити ці 3 задачі максимально точно, не дивлячись на їх складність, адже людина здатна навчити обчислювальну машину визначати переможців та рекордсменів, запобігати зіткненням та враховувати більшість чинників, які заважали б людям, але не обчислювальній машині.

Щодо систем відеоспостереження, тут також є певний спектр задач, пов’язаний з велосипедним транспортом, які варті вирішення. Це дорожньо-транспортні пригоди за участю автомобільного та велосипедного транспорта, небезпечні ситуації, викрадення велосипедного транспорту. Технології комп’ютерного зору здатні попередити про ці випадки для подальшого запобігання з боку людей.

1.2 Модель розпізнавання велосипедного транспорту

Для нашого майбутнього застосунку, а саме частини де буде відбуватися розпізнавання, варто проаналізувати існуючі моделі та обрати ту, яка для досягнення цілей нашої роботи підійде найбільше.

Існує такі найпоширеніші моделі, що дозволяють виконати задачу детектування потрібних об’єктів:

1. Faster R-CNN

2. You Only Look Once (YOLO)

Розглянемо кожну з них окремо та почнемо з Faster R-CNN. Дана модель є продовженням архітектури R-CNN (Region-Based Convolutional Neural Network, регіональна згорткова нейронна мережа), що була розроблена ще у 2014 році Росом Гіршиком. Принцип роботи її архітектури полягає у наступному:

1. Зображення подається на вхід згортковій нейронній мережі і таким чином формується карта ознак, яка оброблюється прошарком RPN.
2. Ковзне вікно проходить по карті ознак, центр ковзного вікна пов’язаний з центром якорів, що є областями, маючими різні співвідношення сторін та розміри.
3. Використовується 3 співвідношення сторін та 3 розміри, на основі метрики IoF (intersection-over-union), степені пересічення якорів та істинних розмічених прямокутників, виноситься рішення про поточний регіон – є об’єкт чи немає.
4. Далі використовується алгоритм FastCNN: карта ознак з отриманими об’єктами передаються прошарку RoI (Region of Interest) з подальшою обробкою повнозв’язних прошарків та класифікацією, а також з визначенням зміщення регіонів потенційних об’єктів.

Наступною моделлю, яку ми розглянемо, є YOLO (You Only Look Once, Варто Лише Одного Разу Поглянути). YOLO є сучасним алгоритмом глибокого навчання, що був розроблений Джозефом Редмоном та Алі Фархаді в 2016 році.

Відмінність YOLO від інших алгоритмів згорткової нейронної мережі в тому, що дана модель швидко розпізнає об’єкти в режимі реального часу. Принцип її роботи полягає у введенні відразу всього зображення, що проходить через згорткову нейронну мережу лише один раз, тобто саме тому вона і має назву «Варто лише одного разу поглянути», адже в інших моделях цей процес проходить з багатократними ітераціями, що сповільнює розпізнавання.

Алгоритм YOLO вже навчений на певному типі набору даних, що складається з 80 різноманітних типів класів (рис.1.3).

person	bird	suitcase	fork	chair	toaster
bicycle	cat	frisbee	knife	sofa	sink
car	dog	skis	spoon	pottedplant	refrigerator
motorbike	horse	snowboard	bowl	bed	book
aeroplane	sheep	sports ball	banana	diningtable	clock
bus	cow	kite	apple	toilet	vase
train	elephant	baseball bat	sandwich	tvmonitor	scissors
truck	bear	baseball glove	orange	laptop	teddy bear
boat	zebra	skateboard	broccoli	mouse	hair drier
traffic light	giraffe	surfboard	carrot	remote	toothbrush
fire hydrant	backpack	tennis racket	hot dog	keyboard	
stop sign	umbrella	bottle	pizza	cell phone	
parking meter	handbag	wine glass	donut	microwave	
bench	tie	cup	cake	oven	

Рисунок 1.3 – набір даних, на якому навчений алгоритм YOLO

YOLO має здатність виявляти всі ці 80 видів об’єктів на зображенні, проте даний алгоритм також може бути спеціально навчений для знаходження нових об’єктів.

В чому ж полягає принцип роботи YOLO? Спочатку зображення, що вводиться у згорткову нейронну мережу, розділяється на секції. Візьмемо для прикладу матрицю-сітку 3*3 (рис.1.4).

Рисунок 1.4 – зображення, що розділене сіткою 3*3

Кожна з секцій на зображенні має певні параметри. Якщо ми припустимо, що загальна кількість класів, які ми шукаємо на малюнку, рівна 3 (скоріше за все це людина, автомобіль та літак), то кожна секція в загальній складності містить 5 параметрів та три параметри класу.

Для коментування цих параметрів, нам потрібно розуміти що таке обмежуючі рамки. При підготовці навчальних даних ми повинні виділити об’єкт, який хочемо виявити на зображенні, тому ми це робимо за допомогою обмежуючих рамок, які як правило представляють з себе квадрати або прямокутники, що виділяють певну частину зображення, в нашому випадку – це автомобілі, що присутні на зображенні (рис.1.5).

Рисунок 1.5 – обмежуючі рамки на автомобілях, що є на зображенні

Тепер нам необхідно дізнатися значення 5 параметрів в кожній секції в матриці 3x3. Нижче представлена світлина окремої секції, в якій є автомобіль.

Рисунок 1.6 – окрема секція з автомобілем

Червона точка посередині позначає центр обмежуючої рамки, горизонтальна синя стрілка – параметр tx (відстань між червоною точкою та найлівішою частиною цієї секції), вертикальна синя стріка – це ty (відстань між червоною точкою і вершиною секції), горизонтальна біла стрілка – це ширина обмежуючої рамки по відношенню до секції, вертикальна біла стрілка – висота обмежуючої рамки по відношенню до секції.

Також присутні такі значення, як індекс об'єктності, що вказує на наявність обличчя в обмежуючій рамці; індекси p1, p2, p3 вказують на ймовірність, що цей об'єкт виявиться людиною, автомобілем або літаком відповідно. Всі 9 секції, присутніх у матриці 3*3, мають ці 8 параметрів і саме вони допомагають алгоритму YOLO точно виявити об'єкт.

Проаналізувавши ці дві моделі, потрібно обрати одну з них, яку ми використовуватимемо в нашій системі розпізнавання велосипедного транспорту. Так як нам потрібне оперативне реагування на можливі нещасні випадки, їх запобігання та є потреба у досягненні інших цілей, що потрібні «тут і зараз», нами було обрано алгоритм YOLO, адже на відміну від Faster R-CNN він виконує поставлену задачу набагато швидше.

1.3 Постановка задачі

Зазвичай розробники вирішують задачі з розпізнавання, відстежування звичайного автомобільного транспорту. Це є найпоширенішою практикою, адже кількість дорожньо-транспортних пригод не меншає, а тому інструменти які здатні на них оперативно реагувати або сприяти їх запобіганню потрібні як ніколи, бо від цього залежить життя як водіїв та їх пасажирів, так і звичайних перехожих, тобто усіх учасників дорожнього руху.

За певних причин, під час вирішення таких задач оминається велосипедний транспорт, хоча наразі спостерігається тенденція переходу на нього через екологічність, можливість зекономити та більш корисний вплив на здоров'я кінцевого користувача. Якщо ми кажемо про дорожньо-транспортні пригоди, то

вони можуть бути не лише за участі автомобільного та велосипедного транспорту, а також за участі лише велосипедистів.

Окрім цього, розпізнавання велосипедного транспорту допоможе запобігти нещасним випадкам під час змагань з певних видів спорту, де застосовуються велосипеди, а також є можливість відстежування транспорту задля об'єктивного оцінювання майстерності учасників змагань.

Тобто, задача розпізнавання транспорту може допомогти вирішити одразу декілька інших задач, які також важливі суспільству.

Розробляючи інформаційну систему розпізнавання велосипедного транспорту, слід пам'ятати основні принципи створення таких систем:

1. Врахування стандартних випадків, пов'язаних з велосипедним транспортом

2. Точність розпізнавання велосипедного транспорту та його виокремлення від інших об'єктів в кадрі

3. Оперативність розпізнавання велосипедного транспорту

Щодо першого пункту, типовими випадками можуть бути:

- зіткнення двох або більше водіїв велосипедних транспортів, проте також варто враховувати що вони можуть рухатися паралельно між собою і не кожне пересічення можна вважати як потенційну дорожньо-транспортну пригоду;
- зайві об'єкти в кадрі, що ускладнюють вирішення задачі розпізнавання транспорту;
- декілька одиниць велосипедного транспорту, що рухаються дотично одне до одного, проте не створюють аварійної ситуації (даний випадок є актуальним для змагань на велотрекінгових доріжках).

Також для кожного з цих випадків варто враховувати другий принцип з перелічених, а саме точність розпізнавання. При відсутності точності, система може розпізнати будь-яке пересічення як дорожньо-транспортну пригоду або

взагалі не розпізнати окремо кожний велосипедний транспорт, що знаходиться в кадрі.

Щодо третього принципу, він гратиме роль як під час стеження у вуличних умовах, так і під час велосипедних змагань. Низька швидкість ознайомлення з ситуацією унеможливлює запобігання нещасним випадкам, що може привести до неприємних ситуацій для учасників дорожнього руху або змагань.

2 ОГЛЯД АРХІТЕКТУРИ ТА ВИБІР МЕТОДІВ РЕАЛІЗАЦІЇ

2.1 Алгоритми комп’ютерного зору

Перш ніж розглядати відомі алгоритми комп’ютерного зору, потрібно розібратися що взагалі з себе представляє комп’ютерний зір як такий.

Комп’ютерний зір (Computer vision, CV) – це область машинного навчання та комп’ютерних наук, що допомагає обчислювальним машинам розуміти світ шляхом розпізнавання візуальних образів та виявлення об’єктів, як це відбувається у людей. Технологія є одним з підрозділів штучного інтелекту.

Місією комп’ютерного зору є навчання обчислювальної машини бачити та розуміти оточення за допомогою цифрових фотографій та відеозаписів, а для досягнення цієї цілі використовуються три компоненти:

1. Отримання зображень. Процес приведення світу до цифрового вигляду, для чого використовується обладнання на кшталт веб-камер, цифрових або зеркальних фотоапаратів, а також професійні 3D-камери.
2. Обробка інформації. Даний компонент необхідний для визначення точок, сегментів та границь, що є звичайними геометричними фігурами. В ньому використовуються складні математичні алгоритми, а популярними методами обробки інформації в розрізі комп’ютерного зору є виділення границь (edge detection), сегментація, класифікація та виявлення об’єктів.
3. Аналіз даних. Останній крок в комп’ютерному зорі, що дозволяє обчислювальній машині приймати власні рішення за допомогою високорівневих даних, які були отриманні під час обробки інформації. Прикладом високорівневого аналізу є розпізнавання або відслідковування об’єктів.

Розберемо окремо компонент «Обробка інформації», а саме його популярні методи обробки:

1. Визначення границь (edge detection). В даному методі обробки аналізується певна світлина та переводиться в набір вигнутих відрізків та ліній. Цей метод використовується з метою виділення найважливіших частин зображення, що дозволяє зменшити кількість оброблюваних даних.

Рисунок 2.1 – оригінальна світлина (зліва) та результат її обробки за допомогою edge detection (справа)

2. Сегментація. Цей метод використовується для визначення місцезнаходження об’єктів та границь на зображеннях. У процесі обробки алгоритм присвоює мітку кожному пікселю, в результаті чого виходить набір сегментів, що охоплює всі частини зображення або вилучені з нього контури.
3. Класифікація. Даний метод лежить в основі іншого, більш складного алгоритму в комп’ютерному зорі – виявлення об’єктів, що дозволяє відрізнити на одному зображенні, наприклад, кота від собаки та інших відомих йому предметів.

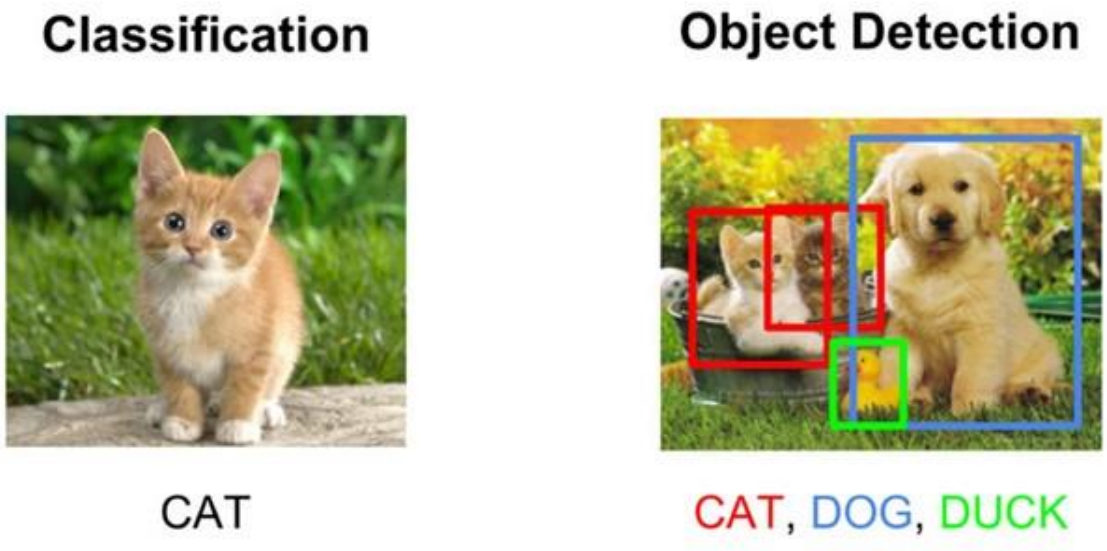

Рисунок 2.2 – класифікація (зліва) та виконання задачі детектування різних об’єктів (справа)

Розглянемо також деякі з інших алгоритмів комп’ютерного зору, що можуть застосовуватися при вирішені певних задач. Такими алгоритмами є каскади Хаара, блендінг, лінійна фільтрація зображень.

Каскади Хаара дозволяють оптимально вирішити задачу детектування обличь. Застосовуючи даний метод, із зображення можна виокремлювати цілком прості ознаки з використанням декількох прямокутників.

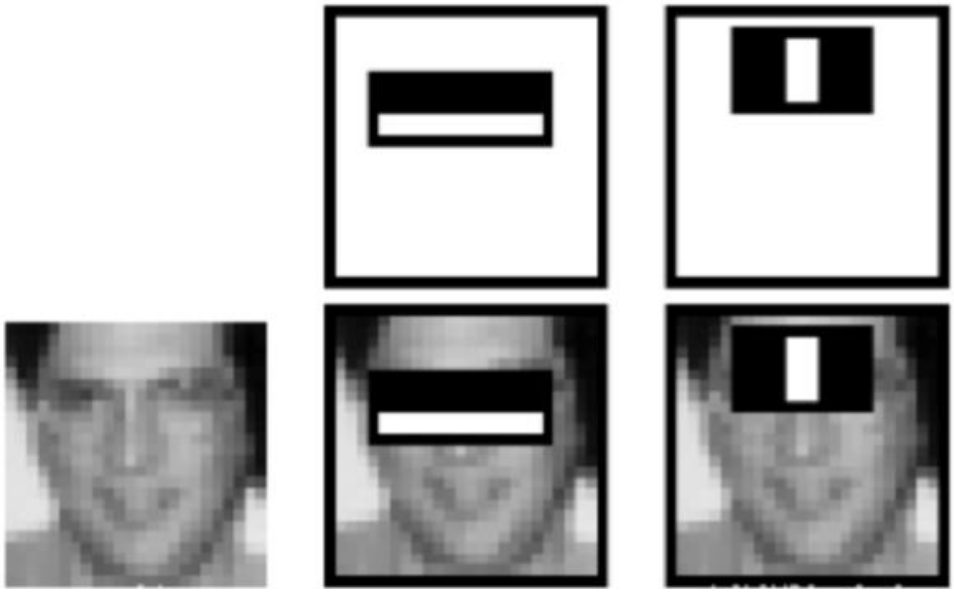

Рисунок 2.3 – наглядний приклад вирішення задачі детектування обличчя за допомогою каскадів Хаара

Пікселі, що потрапляють у білий прямокутник, беруться зі знаком «плюс», а в той же час у чорний зі знаком «мінус». Всі ці значення підсумовуються, виходить одне число. Прямокутники та коефіцієнти для них вибираються за допомогою алгоритму AdaBoost. У обличчя присутні деякі патерни, у підсумку каскад фільтрів такого типу відображає, чи є в ньому обличчя, чи ні.

Звичайно, що існують вже більш досконалі методи детектування обличь, на відміну від каскадів Хаара, проте даний метод є найпростішим, що зустрічається вже у готовому вигляді.

Тепер розглянемо блендінг. Він є прикладом застосування простих арифметичних операцій до зображень. Нехай для одного зображення відомо місцерозташування об'єкту, а весь інший простір займає фон. Тоді можна розміщувати другий об'єкт туди, де знаходиться фон, а у місці де перший об'єкт накривається другим, буде так само використовуватися другий об'єкт.

Таке об'єднання вимогливе до якості вирізання зображення: якщо по краям необережно обрізаний фон, то буде помітною неякісна біла лінія.

Фон є неоднорідним, а тому просто вилучити з зображення білі пікселі не вдасться і тут доведеться піти на хитрість, а саме змішати два зображення та побудувати маску таким чином, що її значення буде тим більше, чим далі піксель від білого. Там, де на вихідному зображенні розташовані білі плями, беруться пікселі з другого зображення і неякісне вирізання зображення буде менш помітним.

Рисунок 2.4 – приклад розпізнавання об’єктів за допомогою блендінгу

Останнім алгоритмом, який буде розглянуто в даному розділі, є лінійна фільтрація зображень. За допомогою лінійних фільтрів вирішуються задачі пошуку границь, кутків, видалення шумів.

Найпростіше пояснити лінійну фільтрацію можна за допомогою певного прикладу. Нехай потрібно підрахувати середнє арифметичне у вікні 3x3 для кожного пікселя. Обрахування середнього арифметичного можна записати так:

$$g[n,m] = \frac{1}{9}\sum_{k=n-1}^{n+1}\sum_{l=m-1}^{m+1} f[k,l] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k, m-l]$$

Переписав формулу інакше, можна отримати формулу для згортання:

$$(f * h)[m,n] = \frac{1}{9}\sum_{k,l} h[m-k, n-l]$$

де f – це зображення (двовимірна функція, характеризуюча зображення); k, l – координати пікселя; f – яскравість пікселя; h – ядро згортання (матриця 3x3, що складається з одиниць).

h

$$1/9 \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Якщо ядро згортання це матриця, то згортання – це плаваюче середнє. За допомогою інструменту OpenCV таке згортання можна провести наступним чином:

```
kernel = np.array([[1,1,1],[1,1,1],[1,1,1]], np.float32) / 9
dst = cv2.filter2D(img1, -1, kernel)
```

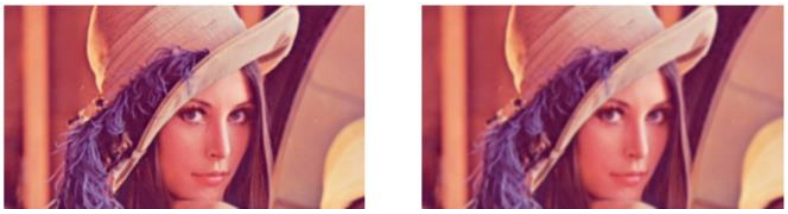

Рисунок 2.4 – світлина до застосування згортання (зліва) та після його застосування (справа)

Отже, ми розглянули відомі алгоритми комп’ютерного зору і можемо перейти безпосередньо до аналізу відомих інструментів для вирішення нашої задачі.

2.2 Аналітичний огляд інструментів комп’ютерного зору

Для вирішення задачі, пов’язаної з розпізнаванням велосипедного транспорту, є потреба у використанні інструментів для комп’ютерного зору. В даній роботі не будуть розглядатися комерційні продукти, але це не вплине на якість виконання задачі, адже рішення з відкритим вихідним кодом дозволяють вирішувати задачі на належному рівні.

Одним з таких рішень є OpenCV, що є бібліотекою з алгоритмами комп’ютерного зору, обробки зображень та функцій, надає інструментарій для аналізу та обробки зображень, вирішення відомих задач (розпізнавання, відстеження, перетворення, застосування методів машинного навчання).

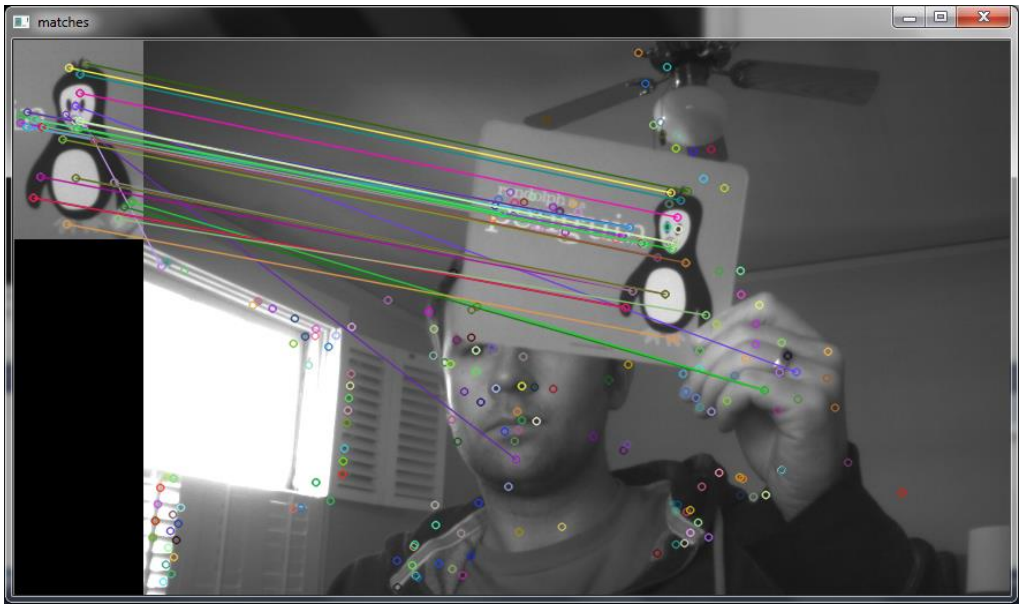

Рисунок 2.5 – приклад роботи OpenCV

Дана бібліотека розроблена компанією Intel, проте наразі підтримується компаніями Willow Garage та Itseez. Не дивлячись на те, що вихідний код OpenCV написаний мовою C++, існують адаптації для таких мов програмування, як Java, Ruby, Matlab, Lua, Python та інших.

Для обрання наступних інструментів вирішення нашої задачі, потрібно визначитися з мовою програмування, яку ми використаємо безпосередньо для розробки продукту, що розпізнаватиме велосипедний транспорт.

Враховуючи, що в рамках цієї роботи буде розроблено першопочаткову версію продукту, яка дозволить ознайомитися з принципами розпізнавання, відстеження велосипедного транспорту, для цієї задачі ідеально підходить мова програмування Python.

Python являє собою інтерпретовану об’єктно-орієнтовану мову програмування високого рівня, що була розроблена в 1990 році Гвідо ван Россумом. Дана мова програмування широко використовується у більшості випадках (веб-застосунки, тестування, обчислювання великого масиву даних, менеджмент тощо) і також чудово підходить для алгоритмів комп’ютерного зору та машинного навчання.

Python пропонує розробникам безліч бібліотек та фреймворків для вирішення подібних задач. Наприклад, PyTorch – відкритий фреймворк для машинного навчання на основі бібліотеки Torch, що розроблена мовою Lua.

Даний фреймворк забезпечує такі високорівневі функціональності:

1. Тензорні обчислення з прискоренням через графічні процесори.
2. Глибинні нейронні мережі, що побудовані на основі системи автоматичного диференціювання на основі плівки.

Для вирішення нашої задачі ми також використаємо imutils – набором функцій, що дозволяє проводити базову обробку зображень (зміна розміру, розташування, визначення границь тощо). У комбінації з OpenCV дана бібліотека спрощує роботу з більшістю алгоритмів і це нам також дуже потрібно.

Отже, набором інструментів, що буде використано для вирішення нашої задачі, є:

1. Мова програмування Python
2. Бібліотека OpenCV
3. Набір утиліт imutils

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель

Продумуючи структуру нашого застосунку для розпізнавання велосипедного транспорту, потрібно врахувати наступні чинники:

1. Система має здатність до масштабування у подальшому
2. Інформаційна система зрозуміла для кінцевого користувача
3. Є можливість для інтеграції в належну інфраструктуру

При проектуванні застосунку потрібно врахувати наявність таких компонентів:

1. Безпосередньо сам застосунок.
2. Система розпізнавання, яка є ядром нашого застосунку та виконує задачі розпізнавання та відстежування об'єктів.
3. Датасет, за яким працює система розпізнавання.
4. Конфігурації, за якими система розпізнавання виконує свою роботу.

На рисунку 3.1 зображено схематичне представлення застосунку.

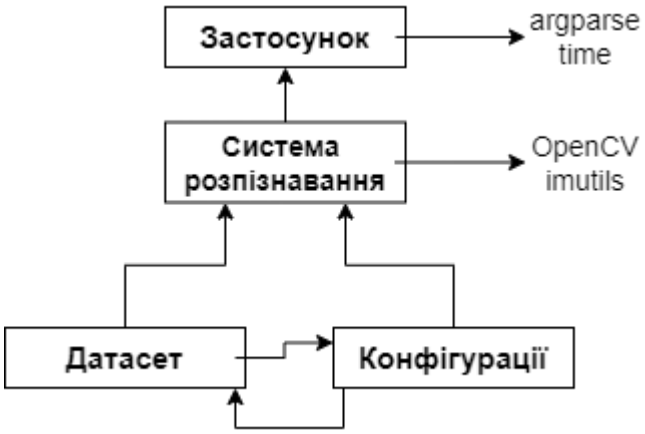

Рисунок 3.1 – схематичне представлення майбутнього застосунку

Зупинимось окремо на датасеті. Перш ніж до нього перейти, варто розібратися з його визначенням. Датасетом називають набір даних, що використовується для аналізу та технологій машинного навчання.

Розглянемо на простому прикладі, чого саме він є важливим для даних технологій. Напередодні того, як їхати в ліс по гриби, ми повинні розуміти, які

гриби є їстівними, а які – отруйними. Найоптимальнішим способом їх розрізнення – за зовнішнім виглядом, але для цього ми повинні розуміти як різні види грибів виглядають. Для цього ми знаходимо зображення в інтернеті або беремо книжку та починаємо переглядати, які види нам варто збирати, а яких краще уникати.

Датасет для нашого застосунку працює за схожим принципом: ми збираємо велику кількість медіаматеріалів з різним освітленням, ракурсами, зайвими об’єктами та навчаємо нашу систему розпізнавання знаходити на світлинах або відео велосипедний транспорт, як саме він пересувається, які випадки є наближеними до аварійних ситуацій тощо.

На рисунку 3.2 представлено фрагмент нашого датасету для системи розпізнавання велосипедного транспорту.

Рисунок 3.2 – фрагмент датасету «Змагання з велотреку»

Розглянемо коротко кожний фрагмент даного датасету. Назви, представленні на зображенні, вже мають в собі короткі ознаки того, що на цих фрагментах відбувається. На першому фрагменті ми спостерігаємо 4 велосипедистів в одному діагональному ряді з можливим пересіченням, яке не передбачає аварійної ситуації; на другому фрагменті ми бачимо 6 велосипедистів в лінію, з якими знаходяться тренери, що є «зайвими об’єктами» для нашої системи; на третьому фрагменті ми спостерігаємо 6 велосипедистів в одному

діагональному ряді, що в самому початку містить в собі «зайвий об’єкт» у вигляді мотоцикліста»; на четвертому фрагменті ми бачимо чотири велосипедисти в активному русі, що їдуть по викривленій лінії; на п’ятому фрагменті ми бачимо двох велосипедистів на треку, які мають достатню відстань між собою, а також спостерігаємо «зайвий об’єкт» у вигляді судді; на останньому фрагменті ми бачимо одну з ідеальних ситуацій, коли 3 велосипедисти їдуть в один ряд, але мають між собою достатню відстань для розпізнавання нашою системою.

Тобто, ми розглянули всього лише 6 фрагментів, які хоч і мають щось спільне, але випадки на кожному з них по своєму унікальні. Варто розуміти, що випадків, чинників, ракурсів може бути безліч, а отже цей датасет налічує зовсім не 6 фрагментів. У випадку з нашою роботою датасет налічує 54 світлини та 12 відеозаписів перегонів з різних ракурсів, з різноманітним освітленням, варіаціями розміщення спортсменів на велосипедному треку, а також з нещасними випадками, коли один учасник змагань падав на велосипеді та тягнув за собою одного або декількох інших велосипедистів. Для його покращення, датасет повинен мати більшу кількість матеріалу, але відповідно для цього потрібні витриваліше та потужніше обладнання, що є недоцільним в рамках даної роботи, адже головною задачею даної роботи є MVP (minimal valuable product, мінімально життєздатний продукт).

Повернемось до проєктування, а саме проєктування зв'язку «Користувач ⇔ Застосунок».

При побудові взаємодії між користувачем та системою, треба врахувати наступне:

1. Користувач повинен надати фото- або відеоматеріал для аналізу системою розпізнавання в застосунку.
2. Користувач може надати конфігурації, за допомогою яких застосунок зрозуміє, за яких саме параметрів йому працювати.
3. В сутності користувача може виступати як жива людина, так і інший

застосунок, а отже наш продукт може бути інтегрований в повноцінну інфраструктуру.

На рисунку 3.3 представлено взаємодію «Користувач <=> Застосунок»

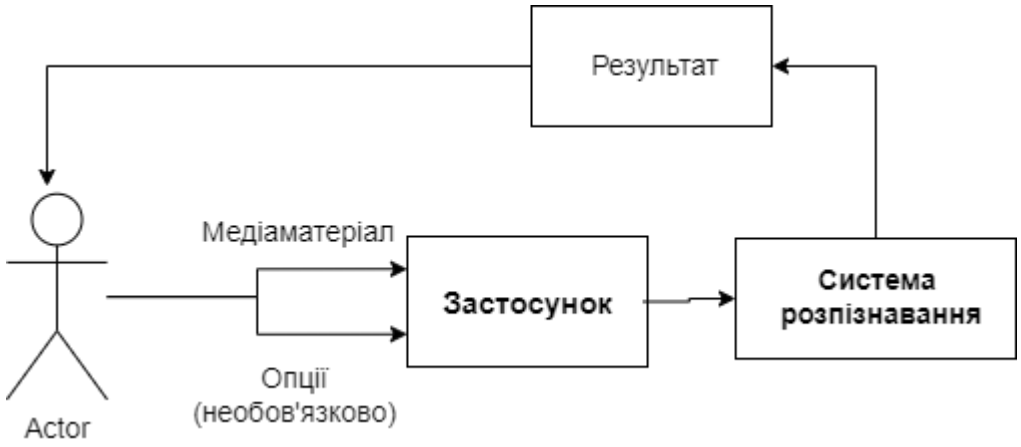

Рисунок 3.3 – схематичне представлення взаємодії «Користувач-Застосунок»

Розробимо базовий алгоритм нашого майбутнього застосунку. При його розробці варто врахувати такі випадки:

1. OpenCV повинен бути новіший за версію 3.2. Якщо ця умова не виконана – застосунок ініціалізує створення власного трекеру. Якщо ця умова виконана – буде ініціалізовано всі існуючі трекери, які можливо використати під час роботи.
2. Користувач повинен надати медіавміст. Якщо цього не відбулося – починається пошук камери, яка буде транслювати медіавміст для нашої системи розпізнавання. Якщо користувач надав фото- або відеоматеріал – починається аналіз вмісту.
3. Якщо це відеоряд, тоді потрібно відслідковувати, чи завершено його програвання задля того, щоб застосунок не працював «просто так». Якщо він завершений, тоді потрібно завершити аналіз та підготувати результати.

На рисунку 3.4 представлено базовий алгоритм роботи нашого застосунку згідно вищезазначених чинників.

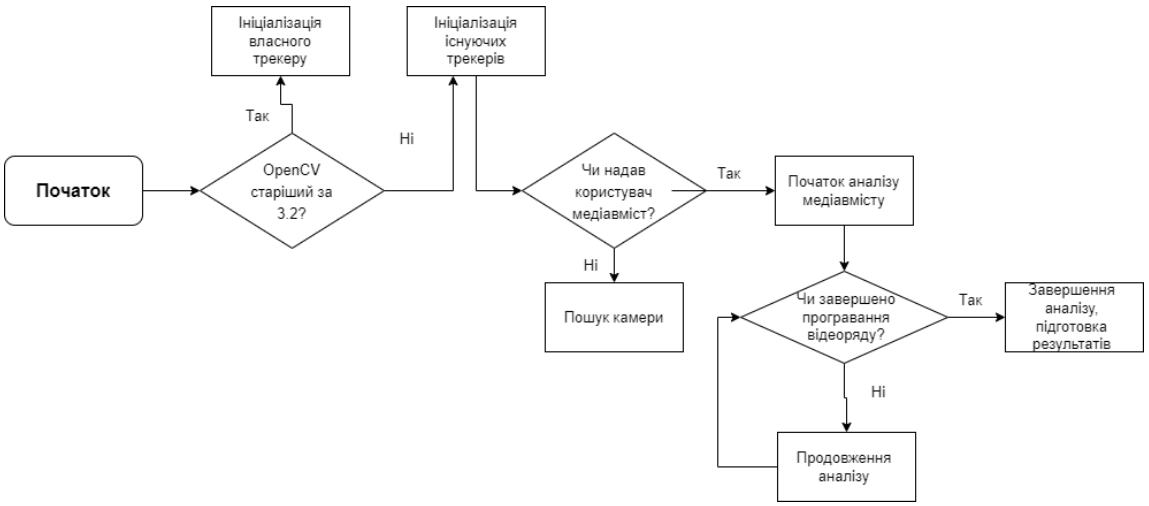

Рисунок 3.4 – схематичне представлення взаємодії «Користувач-Застосунок»

Розробивши структуру та алгоритм нашого застосунку, перейдемо до його розробки та тестування. Вихідний код представлено у додатку А.

3.2 Тестування застосунку

Для тестування нашого застосунку візьмемо запис міжнародних змагань з велотреку у м.Львів (2019 рік). Під час змагань з велотреку, ймовірне скупчення спортсменів, що може привести до нещасних випадків. Проте також є вірогідність, що система розпізнавання може неправильно інтерпретувати дане скупчення. На рисунку 3.5 представлено оригінальний кадр з цих змагань.

Рисунок 3.5 – оригінальний кадр зі змагань з велотреку у м.Львів

На цьому кадрі ми протестували, наскільки чітко система може розпізнати потенційну транспортну пригоду, правильно розставити місця спортсменів та не брати до уваги людину, що переміщається на мотоциклі. На рисунку 3.6 представлено приклад роботи нашої системи.

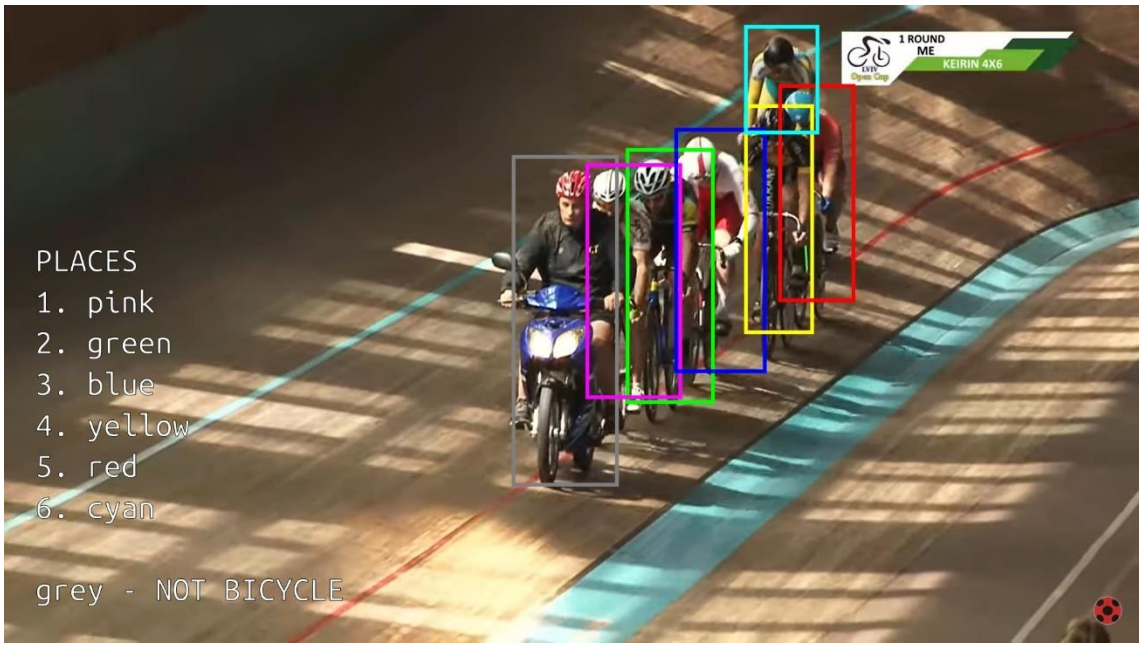

Рисунок 3.6 – той самий кадр зі змагань після аналізу нашою системою

Як ми бачимо, система вірно розподілила місця серед учасників. Окрім цього, вона також виділила організатора, що переміщається на мотоциклі, проте помітила його сірим з позначкою «NOT BICYCLE».

Візьмемо також інший приклад, в якому ми знаємо, що станеться транспортна пригода. На рисунку 3.7 представлено момент напередодні падіння однієї з учасниць змагань в оригінальному вигляді.

Рисунок 3.7 – момент напередодні падіння однієї з учасниць змагань (оригінал)

Спробуємо також проаналізувати цей кадр за допомогою нашої системи. На рисунку 3.8 зображено результат роботи нашої системи розпізнавання.

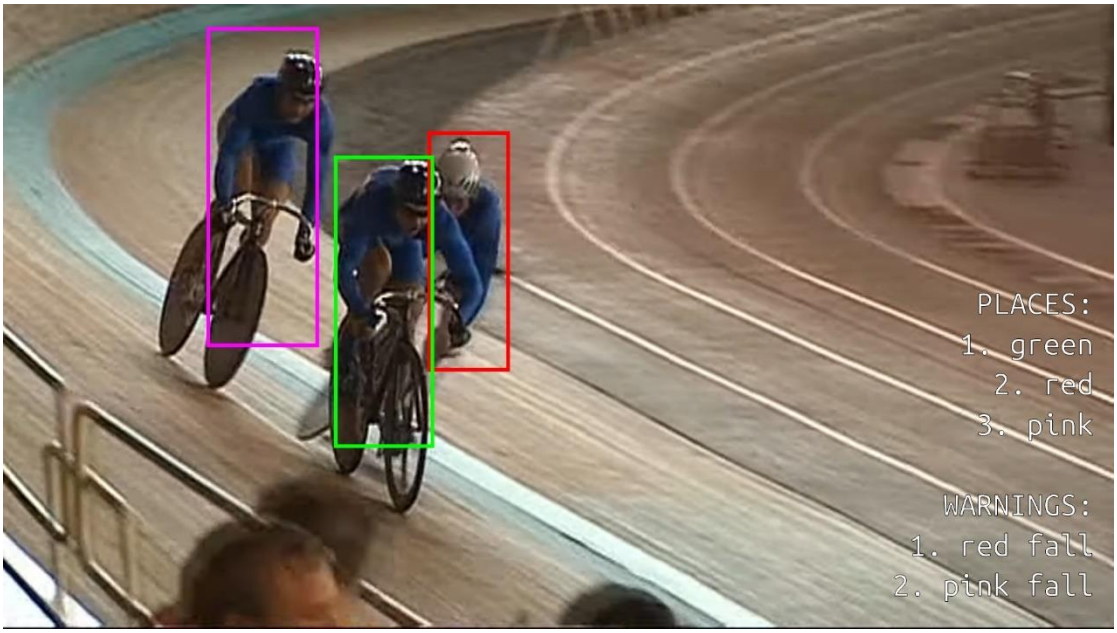

Рисунок 3.8 – момент напередодні падіння однієї з учасниць змагань (аналіз системою)

Як ми можемо помітити, в цьому випадку система також розпізнала коректно всіх учасників змагань та розставила їх позиції на велотреку у правильному порядку. Система також додала два так звані «warnings» (попередження), в яких вказано що велосипедисти red та pink незабаром впадуть.

В цьому є недосконаліть системи, адже на кадрі ми бачимо, що до падіння близька лише велосипедистка red, але не пояснюється чого схожий warning також отримує велосипедистка pink. Проте це є лише візуальним недоліком, адже система розпізнавання прийняла рішення за допомогою критерію Жаккара, який є так би мовити фундаментом під час вирішення задач пересічення обмежувальних прямокутників комп'ютерним зором.

У таблиці 3.1 представлено результати пересічень під час конкретного випадку, що зображений на рисунку вище.

	pink	green	red
pink	-	0.09	0.89
green	0.09	-	0.73
red	0.89	0.73	-

Таблиця 3.1 – коефіцієнти пересічень за критерієм Жаккара

Як ми можемо помітити, пересічення між зеленим та рожевим є мінімальними, адже навіть людським оком помітно що між ними достатньо відстані та аварійної ситуації не передбачено. Між зеленим та червоним пересічення більше, адже при падінні велосипедистка у червоному прямокутнику наблизилася до велосипедистки у зеленому. Залишається відкритим питання – чого між рожевим та червоним найбільший коефіцієнт пересічення? Справа в тому, що під час падіння велосипедистки у червоному прямокутнику, велосипедистка у рожевому не встигла змінити свою траєкторію руху, наблизилася до червоної та зіткнулася з нею, у підсумку також впала.

Не дивлячись на таку недосконаліть, попередження є і вони абсолютно правильні, а отже в подальшому цю систему можна поєднати із системою сповіщення спортсменів, при якій вони зможуть розуміти коли є ймовірність вибути на певний час з перегонів через нещасний випадок.

В перспективі за певними коефіцієнтами критерію Джаккара є можливість передбачати вірогідне зіткнення і, наприклад, при коефіцієнті вище 0.5 за допомогою голосового помічника попереджати спортсмена, що він знаходиться на небезпечній відстані з іншим учасником перегонів.

В подальшому дану систему можна використовувати не лише для перегонів, а також для відслідковування велосипедистів на проїзжій частині, що дозволить запобігати можливим дорожньо-транспортним пригодам або оперативно реагувати відповідним службам, у випадку якщо вона все ж станеться.

ВИСНОВКИ

За результатами аналізу проблеми, було виявлено що майбутній продукт має свою актуальність для її вирішення. Спортивні змагання повинні мати якомога більше автоматизованих процесів за рахунок комп’ютерного зору, а під

час спостереження за проїзжою частиною варто звертати увагу не лише на автомобільний транспорт, а і також на велосипедний. Тим паче, в епоху коли люди активно обирають саме друге.

Було вирішено створити власну систему розпізнавання велосипедного транспорту, яка в автоматизованому режимі дозволяє відслідковувати позиції спортсменів, якщо мова йде про змагання та можливі транспортні пригоди, якщо мова йде також і про спостереження за проїзжою частиною.

У результаті було продумано методи та технології, які були реалізовані для ефективної роботи програмного продукту.

Під час розробки системи розпізнавання були використанні різноманітні технології та знання, зокрема

- основи алгоритмів комп’ютерного зору;
- мова програмування Python;
- бібліотека OpenCV;
- бібліотека imutils;

Розроблений додаток задовольняє всім вимогам, поставленим на етапі постановки завдання.

СПИСОК ЛІТЕРАТУРИ

1. Cascade Classifier [Електронний ресурс] – Режим доступу: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
2. What is Edge detection [Електронний ресурс] – Режим доступу: https://www.mygreatlearning.com/blog/introduction-to-edge-detection/
3. Blending [Електронний ресурс] – Режим доступу: https://homepages.inf.ed.ac.uk/rbf/HIPR2/blend.htm
4. YOLO: You Only Look Once [Електронний ресурс] – Режим доступу: https://laptrinhx.com/yolo-you-only-look-once-2548734518/
5. Semantic Segmentation with Deep Learning [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/semantic-segmentation-with-deep-learning-a-guide-and-code-e52fc8958823
6. Linear Filtering – An overview [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/topics/computer-science/linear-filtering
7. OpenCV Official website [Електронний ресурс] – Режим доступу: https://opencv.org/
8. Python Official website [Електронний ресурс] – Режим доступу: https://www.python.org/
9. PyTorch Documentation [Електронний ресурс] – Режим доступу: https://pytorch.org/docs/stable/index.html
10. Imutils library [Електронний ресурс] – Режим доступу: https://pypi.org/project/imutils/
11. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1506.01497
12. You Only Look Once: Unified, Real-Time Object Detection [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1506.02640
13. What are Convolutional Neural Networks? - IBM [Електронний ресурс] – Режим доступу: https://www.ibm.com/topics/convolutional-neural-networks

ДОДАТКИ

Додаток А. Вихідний код продукту

```
import argparse
import imutils
import time
import cv2

argp = argparse.ArgumentParser()
argp.add_argument("-v", "--video", type=str, help="path to input video file")
argp.add_argument("-t", "--tracker", type=str, default="kcf", help="OpenCV object tracker type")
arguments = vars(ap.parse_args())
(maj, min) = cv2.__version__.split(".")[:2]
if int(maj) == 3 && int(min) < 3:
   trackr = cv2.Tracker_create(arguments["trackr"].upper())
else:
   OCV_TRACKERS = {
      "csrt_tracker": cv2.TrackerCSRT_create,
      "kcf_tracker": cv2.TrackerKCF_create,
      "boosting_tracker": cv2.TrackerBoosting_create,
      "mil_tracker": cv2.TrackerMIL_create,
      "tld_tracker": cv2.TrackerTLD_create,
      "medianflow_tracker": cv2.TrackerMedianFlow_create,
      "mosse_tracker": cv2.TrackerMOSSE_create
   }
   trackr = OCV_TRACKERS[args["tracker"]]()
init_BB = None
if args.get("video", False) != True:
```

```
    print("[INFO] starting video stream...")
    video_stream = VideoStream(src=0).start()
    time.sleep(1.0)
else:
    video_stream = cv2.VideoCapture(args["video"])
fps = None
while True:
    frame = vs.read()
    frame = frame[1] if args.get("video", False) == True else frame
    if frame is None:
        break
    frame = imutils.resize(frame, width=500)
    (H, W) = frame.shape[:2]
    if initBB is not None:


        (success, box) = trackr.update(frame)
        if success:
            (x, y, w, h) = [int(v) for v in box]
            cv2.rectangle(frame, (x, y), (x + w, y + h),
                          (0, 255, 0), 2)
        fps.update()
        fps.stop()
        info = [
            ("Tracker in this case is: ", arguments["trackr"]),
            ("Success?", "Yes!" if success else "No!!!"),
            ("FPS is: ", "{:.2f}".format(fps.fps())),
        ]
        for (i, (k, v)) in enumerate(info):
            text = "{}: {}".format(k, v)
```

```
    cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
if key is ord("s"):
    init_BB = cv2.selectROI("Frame", frame, fromCenter=False,
                            showCrosshair=True)
    trackr.init(frame, init_BB)
    fps = FPS().start()
elif key is ord("q"):
    break
if arguments.get("video", False) != True:
    video_stream.stop()
else:
    video_stream.release()
cv2.destroyAllWindows()
```