Міністерство освіти і науки України

Державний університет «Одеська політехніка»

Інститут штучного інтелекту та робототехніки

Кафедра «Комп'ютерні системи»

Калафицький Артем Дмитрович,

студент групи УК-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ДОСЛІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ СТАНУ ОЧЕЙ ВОДІЯ ДЛЯ УНИКНЕННЯ АВАРІЙНИХ СИТУАЦІЙ

Спеціальність: 123 – "Комп'ютерна інженерія"

Спеціалізація: Спеціалізовані комп'ютерні системи

Керівник:

Ступень Павло В'ячеславович,

кандидат техн. наук, доцент

Одеса — 2021

Міністерство освіти і науки України

Державний університет «Одеська політехніка»

Інститут штучного інтелекту та робототехніки
Кафедра комп'ютерних систем

Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)
Спеціалізація / освітня програма Спеціалізовані ком'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____

“____” _____ 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Калафицькому Артему Дмитровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження методів розпізнавання стану очей водія для уникнення аварійних ситуацій

Керівник роботи Ступень Павло В’ячеславович, кандидат техн. наук, доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ректора ОНПУ від “ 01 ” березня 2021 року № 346-в
2. Зміст роботи
Вступ, Проблеми сонливості водія. Аналіз методів розпізнавання обличчя і машинного зору, Дослідження удосконалення методу спостереження за очима водія, Модулювання системи. Експериментальна перевірка запропонованого методу

3. Перелік ілюстративного матеріалу
Презентація – 14 слайдів

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

5. Дата видачі завдання 01.10.21

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи	01.10.21	Виконано
2	Аналіз предметної області	02.10.21 – 07.10.21	Виконано
3	Аналіз існуючих систем і методів	08.10.21 – 12.10.21	Виконано
4	Дослідження удосконалення методу спостереження за очима водія	13.10.21 – 17.10.21	Виконано
5	Побудова алгоритму роботи методу	18.10.21 – 28.10.21	Виконано
6	Програмна реалізація	29.10.21 – 10.11.21	Виконано
7	Проведення експерименту	11.11.21 – 16.11.21	Виконано
8	Тестування методу та висновки	17.11.21 – 21.11.21	Виконано
9	Оформлення пояснювальної записки	22.11.21 – 30.11.21	Виконано
10	Оформлення презентації	01.12.21 – 11.12.21	Виконано
11	Захист кваліфікаційної роботи	28.12.21	Виконано

Здобувач вищої освіти _____ Калафицький А.Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ступень П.В.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Виявлення сонливості водія - це технологія безпеки автомобіля, яка допомагає запобігти аваріям, спричиненим сонливістю водія. Різні дослідження показали, що близько 20% усіх дорожньо-транспортних пригод пов'язані зі стомлюваністю, до 50% на певних дорогах.

Деякі сучасні системи вивчають поведінку водія та можуть визначати, коли водій стає сонним.

Для виявлення сонливості водія можна використовувати різноманітні технології:

1) Моніторинг моделі кермового управління. В основному використовується кермо від електроенергії рульова система. Цей спосіб спостереження за водієм працює тільки до тих пір, поки водій дійсно активно керує транспортним засобом замість автоматичної системи утримання смуги руху.

2) Положення автомобіля при контролі смуги руху. Використовує камеру спостереження смугою руху. Такий спосіб спостереження за водієм працює лише доти, доки водій активно керує транспортним засобом замість автоматичної системи утримання смуги руху.

3) Контроль очей / особи водія. Використання комп'ютерного зору для спостереження за обличчям водія або за допомогою вбудованої камери, або на мобільних пристроях.

4) Фізіологічні виміри. Потрібні датчики тіла для вимірювання таких параметрів, як активність мозку, частота серцевих скорочень, провідність шкіри, м'язова активність.

Таким чином, удосконалення технологій розпізнавання та запобігання сну за кермом може стати серйозним викликом у галузі покращення систем запобігання аваріям. При виявленні сонливості необхідно в той же момент попередити водія про можливі неприємності. Подібне виявлення досягається за допомогою детектування стану очей водія.

Використання бібліотек мови програмування Python дозволяє виконати програмну реалізацію системи виявлення сонливості водія, що дозволяє визначати, як довго у конкретної людини (водія) були заплющені очі. Якщо очі були закриті протягом певного часу, слід припустити, що водій починає засинати, і включити звуковий сигнал, щоб розбудити водія і привернути його увагу.

Для успішного розпізнавання необхідно розташувати камеру в машині, щоб можна було легко визначити обличчя водія в той момент, коли він знаходиться за кермом, та застосувати локалізацію ознак для спостереження за очима.

Актуальність теми даної кваліфікаційної роботи магістра обумовлена не повною удосконаленостю технологій розпізнавання та запобігання сну водія за кермом у галузі систем запобігання аваріям.

Метою даної кваліфікаційної роботи магістра є розробка і удосконалення методу розпізнавання стану очей водія для уникнення аварійних ситуацій за допомогою алгоритму розпізнавання стану очей.

Завданнями кваліфікаційної роботи магістра є:

1) проаналізувати інформацію існуючих передових методів машинного зору і розпізнавання облич та систем безпеки водія в яких вони використовуються;

2) удосконалити існуючі методи детектування стану облич водія, покращивши швидкодію роботи алгоритму без втрати точності його роботи;

3) практична реалізація пропонованого методу детектування стану очей водія, тестування його на точність роботи і швидкодію;

Практичне значення одержаних результатів: програмна реалізація системи виявлення сонливості водія за допомогою локалізації ознак для спостереження за очима дозволяє визначати, як довго у водія були заплющені очі. Якщо очі були закриті протягом певного часу, слід припустити, що водій починає засинати, і включити звуковий сигнал, щоб розбудити водія і привернути його увагу.

Об’єкт дослідження: процес детектування стану очей водія на зображенні отриманого вбудованою камерою.

Предмет дослідження: методи виявлення розміру форми очей на зображенні отриманого вбудованою камерою.

Методи дослідження: методи досліджень базуються на теорії алгоритмів, методах програмування, методах розпізнавання облич та машинного зору.

Наукова новизна одержаних результатів: вдосконалений метод детектування очей може надійно, точно і в режимі реального часу виміряти візуальну увагу водія до навколишнього середовища, оцінювати ступінь його сонливості і, зрештою, визначати, чи подолав водій поріг ризику. Зниження ризику здійснюється за допомогою інтелектуального оповіщення водія, а також інформування ширших систем керування транспортним засобом. Рішення ненав'язливі, точні, надійні та інтелектуальні. Водію не потрібно нічого носити або змінювати свою поведінку. Технологія не лише відстежує очі, а й вирішує реальні проблеми та рятує життя.

1 ПРОБЛЕМИ СОНЛИВОСТІ ВОДІЯ. АНАЛІЗ МЕТОДІВ РОЗПІЗНАВАННЯ ОБЛИЧЧЯ І МАШИННОГО ЗОРУ

Пропонована система спрямована на удосконалення методів розпізнавання стану очей водія для уникнення аварійних ситуацій. Приведені проаналізовані методи спостереження за обличчям водія мають низку, які потребують удосконалення:

1.1 Проблеми і наслідки сонливості водіїв за кермом

Причиною приблизно 20% усіх серйозних аварій на дорогах є втома водія та, як наслідок, засипання за кермом. Найбільший ризик засипання спостерігається у далеких поїздках, особливо у темний час доби та за монотонних дорожніх умов. Практика показує, що через чотири години безперервного водіння реакція водія знижується вдвічі, через вісім годин – у шість разів. У поїздах і літаках рух все ж таки контролюється автоматикою, а навіть кілька секунд сну на жвавій трасі може призвести до смертельного результату.

При ДТП, зумовлених засипанням за кермом, як зображено на рисунку 1.1, найчастіше відзначаються смертельні наслідки та тяжкі травми. Це обумовлено нездатністю водієм вжити заходів щодо зниження швидкості або ухилення від перешкоди. А з поширенням останнім часом автоматичних коробок передач, водієві просто достатньо заснути і щоб його нога впала на газ. З боку це видається невмотивованим збільшенням швидкості та виїздом на зустрічну смугу або наїздом на перешкоду[1].

Рисунок 1.1 – Засипання водія за кермом автомобіля

Головний фактор при несподіваному засинанні водія добре відомий – це перевтома, накопичення недосипання чи «невиспанність» через поверхневий сон. Найчастішим розладом, що призводить до денної сонливості, є синдром обструктивного апное сну (СОАС). Його поширеність у людей віком понад 30 років, за статистичними даними, становить 4–7%. При цьому близько 2% всього дорослого населення страждає на тяжку форму хвороби. Таким чином, навіть якщо кожен 10 з них керує автомобілем, то на дороги порани щодня виїжджають сотні тисяч водіїв з важкою денною сонливістю[2].

За даними проведеного Європейського дослідження, епізоди сонливості за кермом протягом 2 років спостерігалися у 17% водіїв, хоча і не завжди вони були пов'язані з ризиком СОАС, як показано на рисунку 1.2.

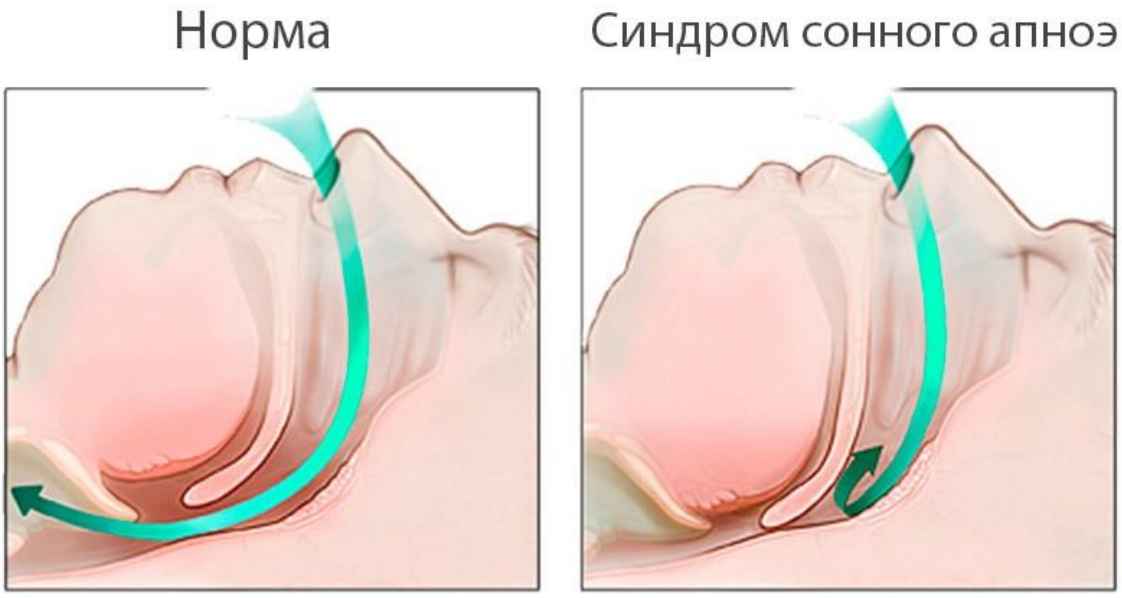

Рисунок 1.2 – Синдром сонного апное при засинанні деяких хворих водіїв, що також призводить до аварій

Приступи засинання – багатофакторний процес, Потрібно обов'язково враховувати психофізіологію людини. Так, астенічні молоді люди, худорляві, ті, хто змушений починати день з міцної кави, схильні до нападів засинання більше за інших. Деякі схильні до засинання після ситної їжі. Наступний фактор ризику - нестиковка, конфлікт біоритмів людини і зміни, в яку йому доведеться працювати. Це стосується і «жайворонків» і так званих «сов». Впливає ергономіка траси, блимання світла або його нестача. Сонливість викликається як втомою, а й хворобами, емоційної пригніченістю, тривалим станом тривоги. В останні роки погану службу грає і комп'ютерна залежність, багато машиністів не висипаються через неї.

СОАС безумовно підвищує ризик дорожньо-транспортних пригод, але надмірна денна сонливість спостерігається лише у 50% таких пацієнтів. Ризик ДТП при СОАС більше пов'язаний зі ступенем денної сонливості, ніж з об'єктивними характеристиками дихальних розладів уві сні – такими, як ІАГ. 25,9% відповідно, що значно перевищувало частоту ДТП у контрольній групі без порушень дихання уві сні.

Система контролю втоми стежить за фізичним станом водія та якщо фіксує певні відхилення, попереджає водія про необхідність зупинки та відпочинку. Залежно від способу оцінки втоми водія розрізняють три типи систем. Перші побудовані на контролі дій водія, другі - на контролі руху автомобіля, треті - на контролі погляду водія, як зображено на рисунку 1.3.

Рисунок 1.3 – Методи спостереження за станом водія

Сонливість характеризується як розумове порушення через нестачу сну. Його неможливо подолати, просто «прокинувшись», єдине рішення – відпочинок. Сонне керування було давньою проблемою безпеки майже на всіх транспортних ринках. На відміну від алкоголю, який можна перевірити на дихання, немає надійного способу виміряти ступінь сонливості водія з будь-яким ступенем впевненості.

Однак після більш ніж десяти років досліджень та використання найкращих у своєму класі базових алгоритмів відстеження голови та погляду у поєднанні з новітніми методами машинного навчання, які застосовуються до ретельно зібраних

наборів даних, компанія Seeing Machines може надавати сигнал рівня сонливості водія (DDL). який пропонує дієву оцінку ризику втрати водієм контролю за транспортним засобом у найближчому майбутньому.

Технології компанії «Бачучі машини» стежать за ознаками того, що водій входить до тями і виходить із нього; так звані події «мікросна».

Деякі водії ніколи не виявляють ознак сонливості або навіть мікросну і можуть просто раптово знепритомніти. Це також може статися з медичних причин, наприклад, через низький рівень цукру в крові. Технологія Seeing Machines здатна виявити водія, який непритомний.

Умовно автономні транспортні засоби, як показано на рисунку 1.4 та розширені функції безпеки, такі як системи запобігання зіткненню, вимагають надійного вимірювання рівня «залучення» водія в завдання водіння (або дорожню сцену). Цей сигнал розроблений, щоб дозволити системам автомобіля розумно та безпечно передати керування або повернути керування від водія[3].

Рисунок 1.4 – Автономне транспортне керування

1.2 Аналіз існуючих систем контролю втоми водія

Інноваційна технологія моніторингу водіїв DS Automobiles випереджає крайній термін безпеки 2022 року, що означає, що нові автомобілі в ЄС повинні бути оснащені системами оповіщення про втому.

Введення нових законів, ухвалених раніше цього року, зробить попередження водіїв про сонливість та відволікання уваги обов'язковим на всіх нових автомобілях у ЄС з 2022 року. З 2015 року у Великій Британії сталося 4000 аварій та 150 смертельних випадків через втому водія.

DS Automobiles випереджає вимоги безпеки завдяки системі DS DRIVER ATTENTION MONITORING, яка допомагає тримати водіїв у напрузі і не дає їм заснути за кермом. Передова технологія відстежує ознаки втоми, що потенційно допомагає врятувати близько 50 життів на рік на дорогах Великобританії.

Система, вже доступна на DS 7 CROSSBACK, поєднує пару інфрачервоних камер, орієнтованих на водія, з безперервним моніторингом положення автомобіля, як зображено на малюнку 1.5.

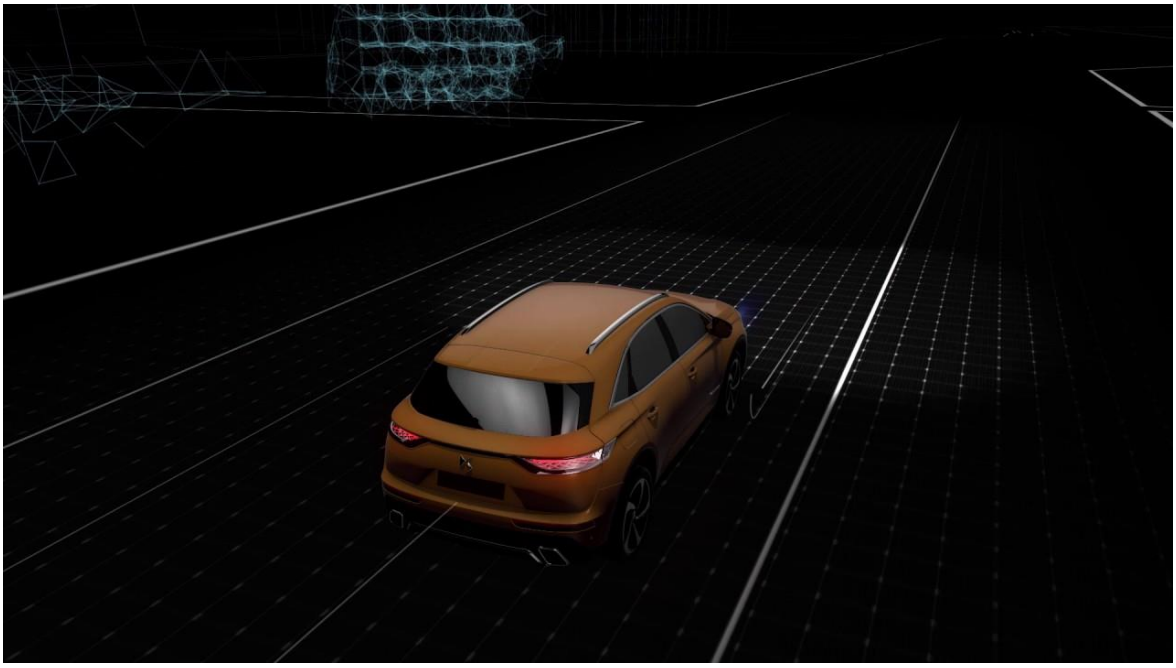

Рисунок 1.5 – Моніторинг положення автомобілю(але за наявністю розмітки)

Футуристичні звернені до водія камери, встановлені над кермовим колесом та у верхній частині лобового скла, відстежують три основні фізичні ознаки відволікання уваги або сонливості; рух очей, повік чи шиї. При виявленні будь-якого з них увімкнено звуковий сигнал, а на дисплеї цифрового приладу з'являється попереджувальне повідомлення.

Водночас система контролю положення транспортного засобу постійно відстежує автомобіль щодо дорожньої розмітки та попереджає водія звуковим сигналом, якщо є якісь раптові чи несподівані рухи кермового керування.

Комбінація цих технологій дозволяє системі DS DRIVER ATTENTION MONITORING всебічно контролювати водіїв на предмет основних запобіжних ознак відволікання уваги та сонливості та максимізувати вікно, в якому водій може бути попереджений та спонуканий вжити заходів для протидії цьому[4].

Cadillac Super Cruise - це нова технологія адаптивного круїз-контролю, зображено на рисунку 1.6. На відміну від Tesla Autopilot та Nissan ProPilot Assist, які в основному покладаються на камери, спрямовані вперед, щоб розпізнавати дорожню розмітку для утримання смуги руху, Cadillac Super Cruise використовує карти високої роздільної здатності для визначення розташування автомобіля з точністю до чотирьох дюймів. Він також знає місцевість попереду відстань до 1,5 миль. Карти HD оновлюються щокварталу повітрям через систему GM OnStar, але можуть надсилатися частіші оновлення, щоб включати зміни умов, таких як зони будівництва.

Рисунок 1.6 - Технологія адаптивного круїз-контролю Cadillac Super Cruise

Хоча автопілот і ProPilot Assist вимагають, щоб руки водія залишалися на кермі, щоб їхня система залишалася включеною, Super Cruise залишатиметься активним лише в тому випадку, якщо водій приділяє увагу. Він визначає це, безперервно відстежуючи напрямок погляду та обличчя водія за допомогою інфрачервоної камери, встановленої на рульовій колонці[5].

Mercedes-Benz з 2011 року встановлює на своїх автомобілях систему Attention Assist, яка зображена на рисунку 1.7, в якій контроль дій водія грунтувався на багатьох факторах: манері їзди, поведінці за кермом, використання органів управління, характері та умовах руху та ін.

Рисунок 1.7 - Система Attention Assist на Mercedes-Benz

Конструкція системи Attention Assist поєднує датчик рульового колеса, блок керування, сигнальну лампу та звуковий сигнал оповіщення водія. Датчик рульового колеса фіксує динаміку дій водія обертанням рульового колеса. У роботі система використовує також вхідні сигнали датчиків інших систем автомобіля: управління двигуном, курсової стійкості, нічного бачення, гальмівної системи.

Блок управління обробляє вхідні сигнали та визначає:

- стиль водіння (аналіз швидкості, поздовжнього та бокового прискорення протягом 30 хв. після початку руху);
- умови керування (аналіз часу доби, тривалості поїздки);
- використання органів керування (аналіз використання гальма, підрульових перемикачів, кнопок на панелі керування);
- характер обертання кермового колеса (аналіз швидкості, прискорення);
- стан дорожнього полотна (аналіз бічного прискорення);
- характер руху автомобіля (аналіз поздовжнього та бокового прискорення).

В результаті проведених обчислень встановлюються відхилення у діях водія та траєкторії руху автомобіля. На дисплеї панелі приладів виводиться сигнальний напис про необхідність зробити перерву та відбувається звуковий сигнал. Якщо

після сигналів водій не зупиняється та продовжує рух у сонливому стані, система повторює сигнали з періодичністю 15 хвилин. Система активується на швидкості 80 км/год.

Система Driver Alert Control, яка зображена на рисунку 1.8, на відміну від системи Attention Assist, система Driver Alert Control, DAC від Volvo фіксує лише характер руху автомобіля дорогою. Спрямована відеокамера вперед фіксує положення автомобіля на смузі руху. Відхилення від заданих параметрів руху розглядається системою як настання втоми водія. Залежно від стану водія в системі реалізовано два рівні попередження - "м'який" та "жорсткий". Рівні відрізняються гучністю та тональністю звукового сигналу. Система DAC працює спільно з системою Lane Departure Warning та базується на її конструктивних елементах. Активація системи відбувається на швидкості 60 км/год.

Рисунок 1.8 - Система Driver Alert Control, DAC від Volvo

Система Seeing Machines - контроль погляду з метою оцінки втоми водія впроваджує компанія General Motors. За основу взято готову технологію Seeing Machines, яка застосовується в авіації, залізничному транспорті, кар'єрних роботах,

комерційному вантажному транспорті. Спеціальний блок контролює ступінь відкриття очей та напрямок погляду водія, рисунок 1.9. При розпізнаванні неуважності, втоми чи сонливості водія система попереджає необхідність зупинки.

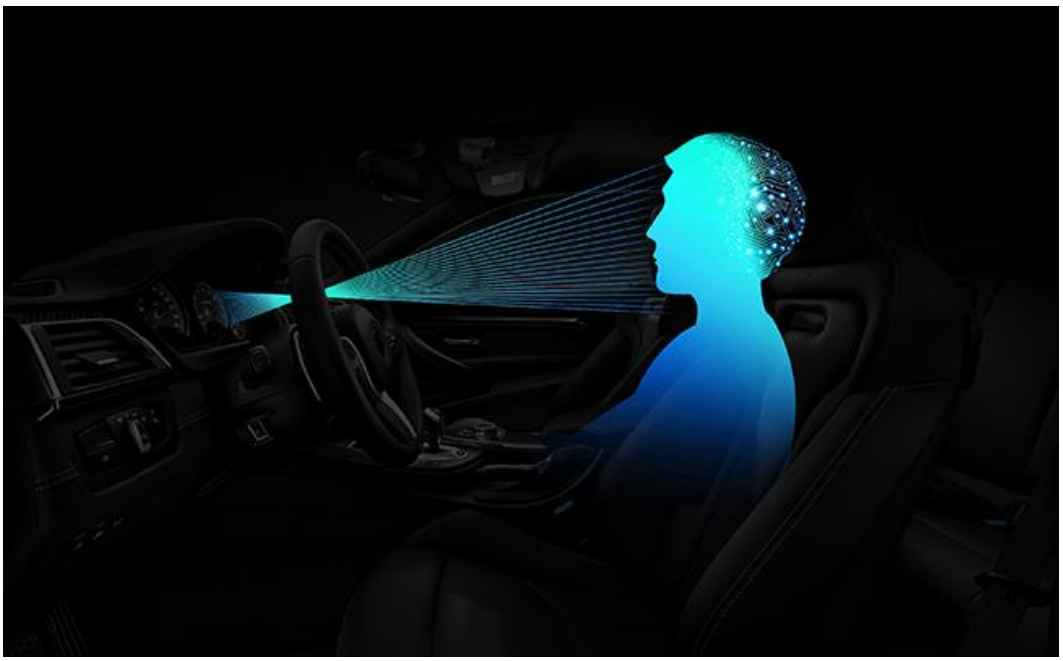

Рисунок 1.9 - Система контролю погляду Seeing Machines

Крім контролю втоми водія, система може бути використана для активації окремих функцій автомобіля за допомогою спрямованого погляду (поглянув - ввімкнув). Крім того, якщо при перестроюванні водій не користується дзеркалом заднього виду, система нагадає йому про необхідність цієї дії[6].

1.3 Аналіз методів розпізнавання обличчя і машинного зору

Розпізнавання облич - це автоматична локалізація людської особи на зображенні або відео і, при необхідності, ідентифікація особи людини на основі наявних баз даних. Інтерес до цих систем дуже великий у зв'язку з широким колом завдань, що вони вирішують[7].

Розпізнавання облич - біометричний програмний додаток, здатний однозначно ідентифікувати або верифікувати людину шляхом порівняння та

аналізу шаблонів на основі контурів особи людини, як зображено на рисунку 1.10 Кожна людина має унікальну будову особи. Спеціальне програмне забезпечення здатне аналізувати його, зіставляючи з інформацією базі даних для подальшої ідентифікації.

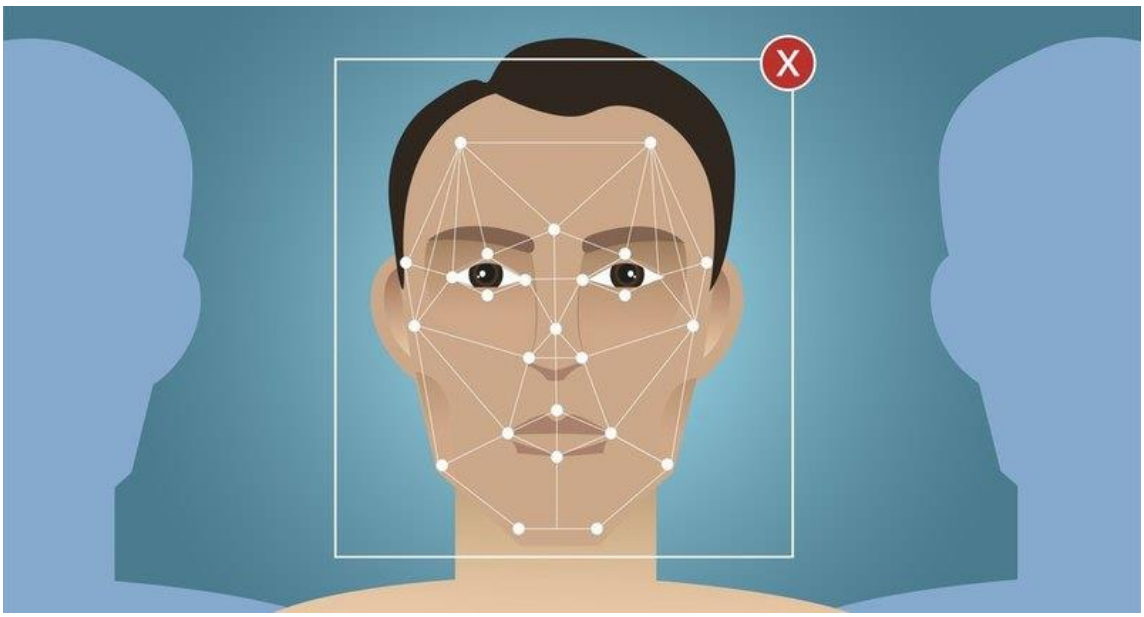

Рисунок 1.10 – Технологія розпізнавання облич

Головний недолік технології розпізнавання облич – це погіршення якості розпізнавання при:

- погіршенні освітленості;
- зміні положення голови чи ракурсу.
- Існує кілька підходів до створення алгоритму розпізнавання осіб.

Емпіричний підхід використовувався на початку розвитку комп'ютерного зору, рисунок 1.11 Він базується на деяких правилах, які використовує людина для визначення особи. Наприклад, лоб зазвичай яскравіший, ніж центральна частина особи, яка, у свою чергу, однорідна за яскравістю та кольором. Ще однією важливою ознакою є наявність частин особи на зображенні - носа, рота, очей. Для визначення осіб проводиться значне зменшення ділянки зображення, де передбачається наявність особи, або будуються перпендикулярні гістограми. Ці

методи легко реалізувати, але вони практично непридатні за наявності великої кількості сторонніх об'єктів на фоні, кількох осіб у кадрі або зміни ракурсу.

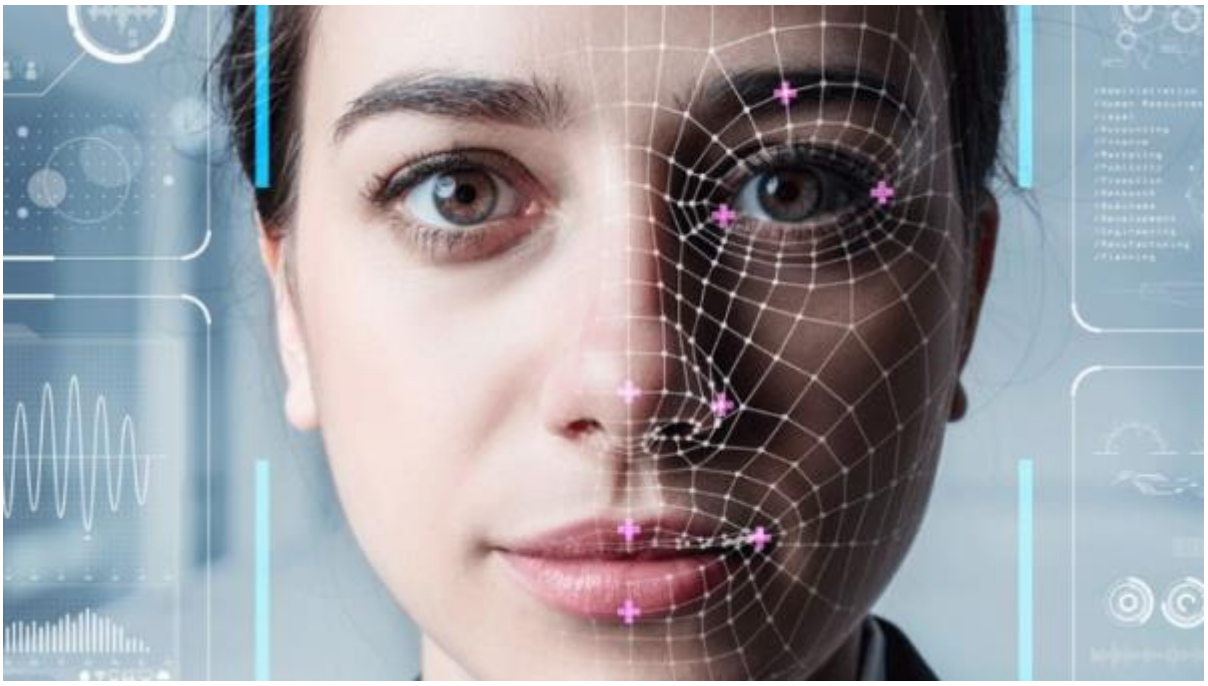

Рисунок 1.11 – Емпіричний метод розпізнавання обличчя

Наступний підхід використовує інваріантні ознаки, характерні зображення обличчя. У його основі, як і попередньому методі, лежить емпірика, тобто спроба системи «думати» як людина. Метод виявляє характерні частини особи, її межу, зміну форми, контрастності тощо, об'єднує всі ці ознаки та верифікує. Даний метод може використовуватися навіть при повороті голови, але за наявності інших осіб або неоднорідному тлі розпізнавання стає неможливим.

Наступний метод – це детектування осіб за допомогою шаблонів, які задає розробник. Особа представляється якимось шаблоном чи стандартом, і мета алгоритму – провести перевірку кожного сегмента наявність цього шаблону, причому перевірка може проводитися до різних ракурсів і масштабів. Така система потребує безліч трудомістких обчислень.

Усі сучасні технології розпізнавання облич використовують системи, які навчаються за допомогою тестових зображень. Для навчання використовуються бази із зображеннями, яка зображена на рисунку 1.12, що містять особи, та не

містять особи. Кожен фрагмент досліджуваного зображення характеризується як вектор ознак, з допомогою якого класифікатори (алгоритми визначення об'єкта у кадрі) визначають, є ця частина зображення обличчям чи ні.

Рисунок 1.12 – База даних зображень для розпізнавання облич

Технологічно системи іноді можуть сильно відрізнятися щодо розпізнавання осіб, але вони мають приблизно загальні принципи роботи.

Незважаючи на велику різноманітність представлених методів, можна виділити загальну структуру процесу розпізнавання облич зображену на рисунку 1.13[8]:

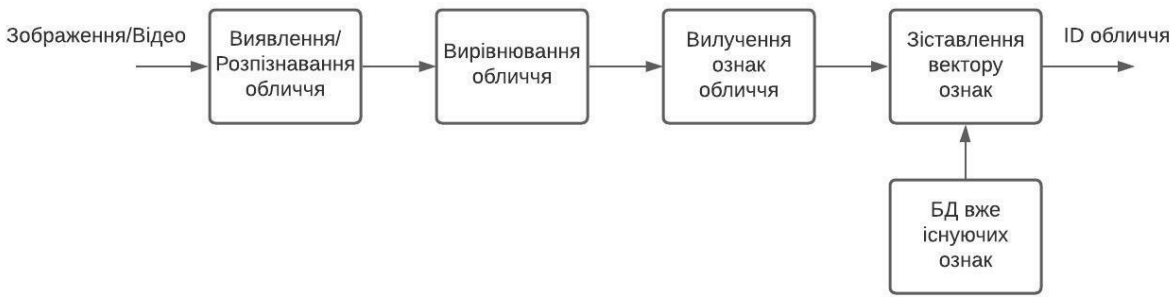

Рисунок 1.13 – Загальна структура процесу розпізнавання облич

1) Зчитування системою зображення/кадра з відеопотоку;
2) Виявлення/Розпізнавання обличчя;
3) Локалізація/Положення обличчя, визначення розміру;
4) Вирівнювання обличчя;
5) Вилучення ознак обличчя;
6) Вилучення ознак з бази даних вже зареєстрованих облич;
7) Зіставлення вектору ознак обличчя з ознаками облич баз даних;
8) Отримання ID обличчя.

На першому етапі проводиться детектування та локалізація особи на зображенні. На етапі розпізнавання виробляється вирівнювання зображення особи (геометричне і яскравість), обчислення ознак і безпосередньо розпізнавання – порівняння обчислених ознак із закладеними базу даних еталонами.

1.3.1 Метод гнучкого порівняння на графах.

Суть методу зводиться до еластичного зіставлення графів, що описують зображення осіб. Особи представлені у вигляді графів зі зваженими вершинами та ребрами. На етапі розпізнавання один із графів – еталонний – залишається незмінним, тоді як інший деформується з метою найкращого припасування до першого. У подібних системах розпізнавання графи можуть являти собою як прямокутні ґрати, так і структуру, утворену характерними (антропометричними) точками обличчя, як зображено на рисунку 1.14.

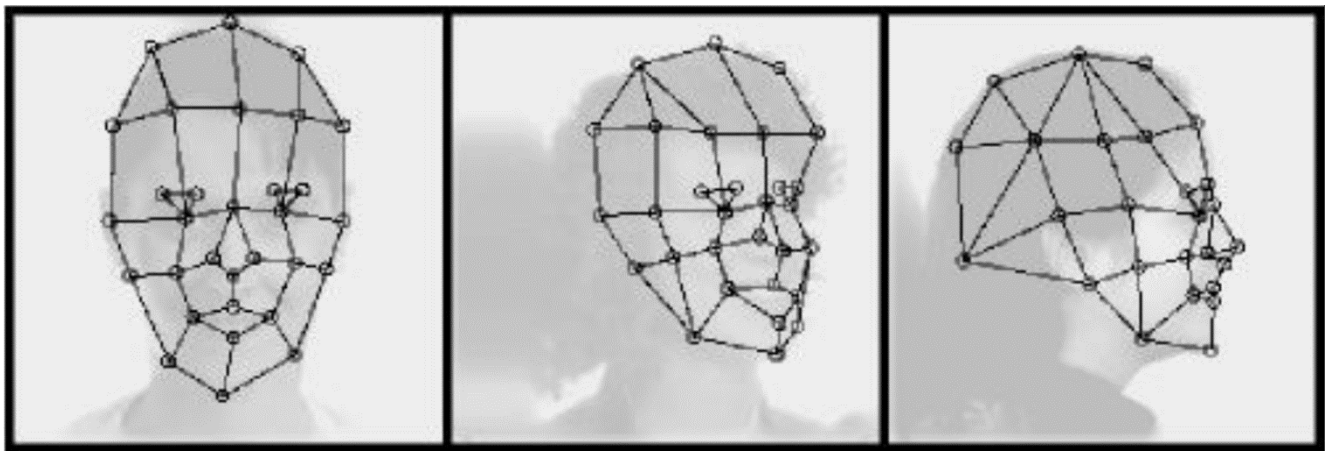

Рисунок 1.14 – Граф на основі антропометричних точок обличчя

У вершинах графа обчислюються значення ознак, найчастіше використовують комплексні значення фільтрів Габора чи його впорядкованих наборів – Габорівських вейвлет (ряди Габора), які обчислюються у певній локальної області вершини графа локально шляхом згортки значень яскравості пікселів з фільтрами Габора.

Ребра графа зважуються відстанями між суміжними вершинами. Відмінність (відстань, дискримінаційна характеристика) між двома графами обчислюється за допомогою деякої цінової функції деформації, яка враховує як різницю між значеннями ознак, обчисленими у вершинах, і ступінь деформації ребер графа. Деформація графа відбувається шляхом зміщення кожної з його вершин на деяку відстань у певних напрямках щодо її вихідного розташування та вибору такої її позиції, при якій різниця між значеннями ознак (відгуків фільтрів Габора) у вершині графа, що деформується, і відповідної їй вершині еталонного графа буде мінімальною. Алгоритм методу гнучкого порівняння наведено на рисунку 1.15.

Рисунок 1.15 - Метод гнучкого порівняння на графах

В окремих публікаціях вказується 95-97% ефективність розпізнавання навіть за наявності різних емоційних виразах і зміні ракурсу обличчя до 15 градусів. Однак розробники систем еластичного порівняння на графах посилаються на високу обчислювальну вартість цього підходу. Наприклад, для порівняння вхідного зображення особи з 87 еталонними витрачалося приблизно 25 секунд при роботі на паралельній ЕОМ з 23 трансп'ютерами.

Недоліки: висока обчислювальна складність процедури розпізнавання. Низька технологічність при запам'ятовуванні нових стандартів. Лінійна залежність часу роботи від обсягу бази даних осіб.

1.3.2 Метод нейронних мереж

В даний час існує близько десятка різновидів нейронних мереж (НМ). Одним із найбільш широко використовуваних варіантів є мережа, побудована на багатошаровому перцептроні, яка дозволяє класифікувати подане на вхід зображення/сигнал відповідно до попереднього налаштування/навчання мережі. Навчаються нейронні мережі з набору навчальних прикладів. Суть навчання зводиться до налаштування терезів міжнейронних зв'язків у процесі розв'язання оптимізаційної задачі методом градієнтного спуску. У процесі навчання НМ відбувається автоматичне вилучення ключових ознак, визначення їх важливості та побудова взаємозв'язків між ними.

Найкращі результати в області розпізнавання облич (за результатами аналізу публікацій) показала Convolutional Neural Network або згорткова нейронна мережа, зображена на рисунках 1.16 та 1.17 (далі – ЗНМ), яка є логічним розвитком ідей таких архітектур НМ як когнітрону та неокогнітрону. Успіх обумовлений можливістю обліку двовимірної топології зображення, на відміну багатошарового перцептрона.

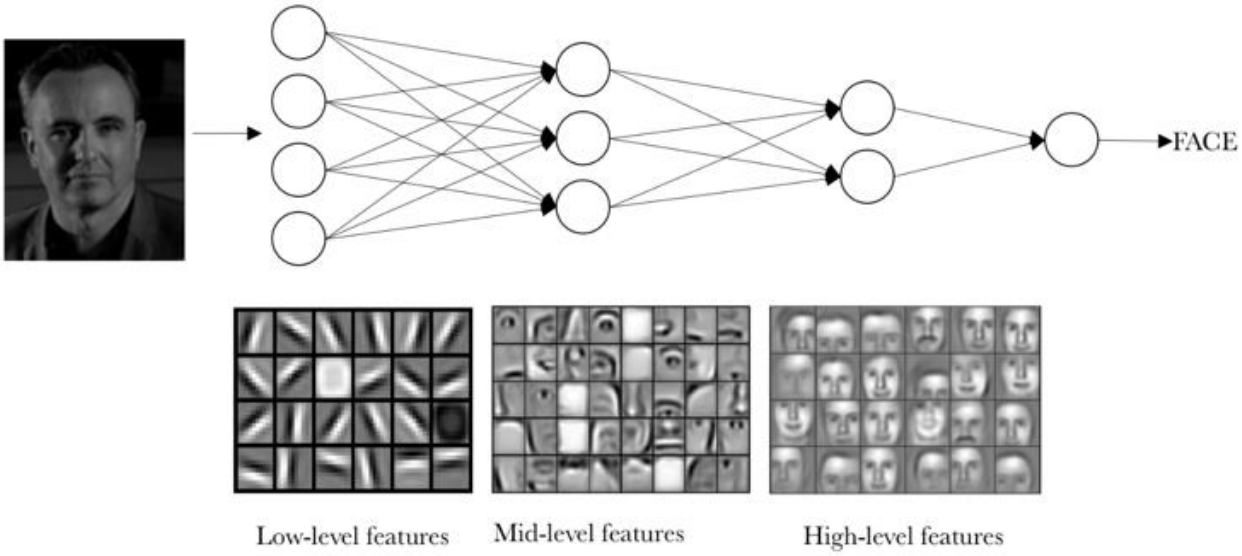

Рисунок 1.16 – Метод згорткової нейронної мережі(ЗНМ)

Відмінними рисами ЗНМ є локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів), загальні ваги (забезпечують детектування деяких рис у будь-якому місці зображення) та ієрархічна організація з просторовими семплінгом (spatial subsampling). Завдяки цим нововведенням ЗНМ забезпечує часткову стійкість до змін масштабу, зсувів, поворотів, зміни ракурсу та інших спотворень.

Рисунок 1.17 - Схематичне зображення архітектури згорткової нейронної мережі

Тестування ЗНМ на базі даних ORL, що містить зображення осіб з невеликими змінами освітлення, масштабу, просторових поворотів, положення та різними емоціями, показало 96% точність розпізнавання. Алгоритм методу

розпізнавання облич на основі загорткових нейронних мереж наведено на рисунку 1.18.

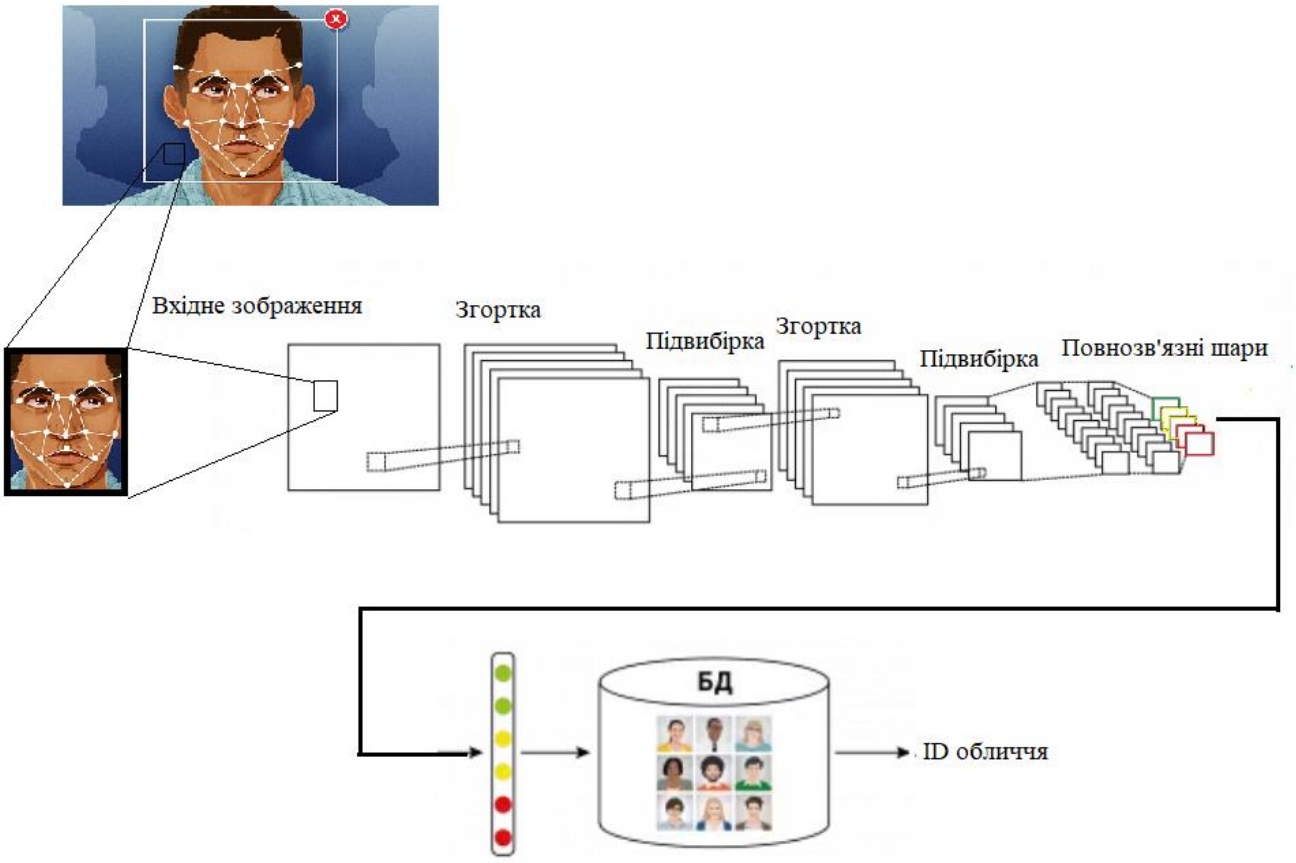

Рисунок 1.18 – Метод розпізнавання на основі загорткової нейронної мережі

Недоліки: додавання нової еталонної особи до бази даних вимагає повного перенавчання мережі по всьому наявному наборі (досить тривала процедура, залежно від розміру вибірки від 1 години за кілька днів). Проблеми математичного характеру, пов'язані з навчанням: попадання в локальний оптимум, вибір оптимального кроку оптимізації, перенавчання.

1.3.3 Сховані Марківські моделі (CMM, HMM)

Одним із статистичних методів розпізнавання облич є приховані Марківські моделі (CMM) з дискретним часом, зображена на рисунку 1.19. CMM використовують статистичні властивості сигналів та враховують безпосередньо їх

просторові характеристики. Елементами моделі є: безліч прихованих станів, безліч станів, що спостерігаються, матриця перехідних ймовірностей, початкова ймовірність станів. Кожному відповідає своя Марківська модель.

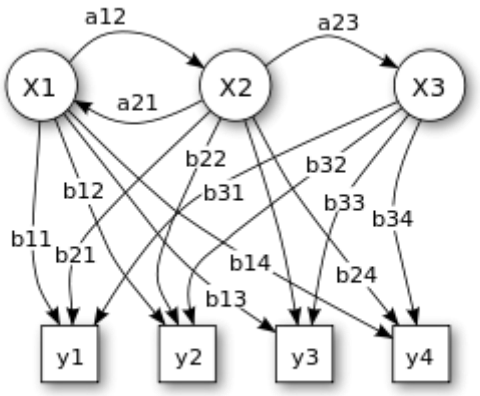

Рисунок 1.19 – Принцип роботи прихованої марківської моделі

При розпізнаванні об'єкта перевіряються згенеровані для заданої бази об'єктів Марківські моделі і шукається максимальна ймовірність того, що послідовність спостережень для даного об'єкта згенерована відповідною моделлю.

Недоліки: необхідно підбирати параметри моделі для кожної бази даних. СММ не має здатності розрізняти, тобто алгоритм навчання тільки максимізує відгук кожного зображення на свою модель, але не мінімізує відгук на інші моделі. На сьогоднішній день не вдалося знайти приклад комерційного застосування СММ для розпізнавання осіб.

1.3.4 Метод головних компонентів або principal component analysis (PCA)

Одним з найбільш відомих і опрацьованих є метод головних компонентів (principal component analysis, PCA), заснований на перетворенні Карунена-Лойова. Спочатку метод головних компонентів почав застосовуватися в статистиці для зниження простору ознак без істотної втрати інформації. У задачі розпізнавання

облич його застосовують головним чином для представлення зображення обличчя вектором малої розмірності (головних компонентів), який порівнюється потім з еталонними векторами, закладеними в базу даних. Принцип роботи методу зображено на рисунку 1.20.

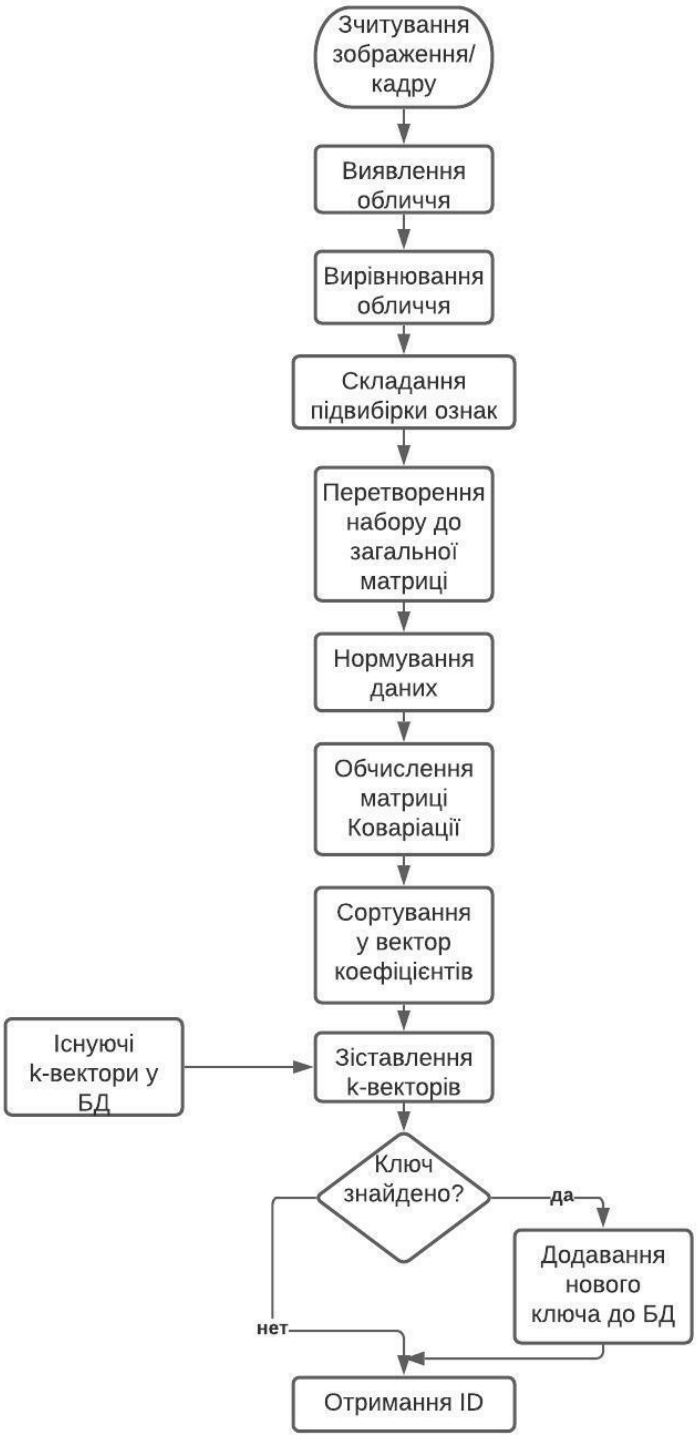

Рисунок 1.20 – Принцип розпізнавання облич методу головних компонентів

Головною метою методу головних компонентів є значне зменшення розмірності простору ознак таким чином, щоб воно якнайкраще описувало «типові» образи, що належать безлічі осіб. Використовуючи цей метод, можна виявити різні мінливості в навчальній вибірці зображень осіб і описати цю мінливість в базі декількох ортогональних векторів, які називаються власними (eigenface).

Суть методу основних компонентів зводиться до наступного. Спочатку весь навчальний набір осіб перетворюється на одну загальну матрицю даних, як зображено на рисунку 1.21, де кожен рядок є один екземпляр зображення особи, розкладеного в рядок. Усі особи навчального набору повинні бути приведені до одного розміру та з нормованими гістограмами.

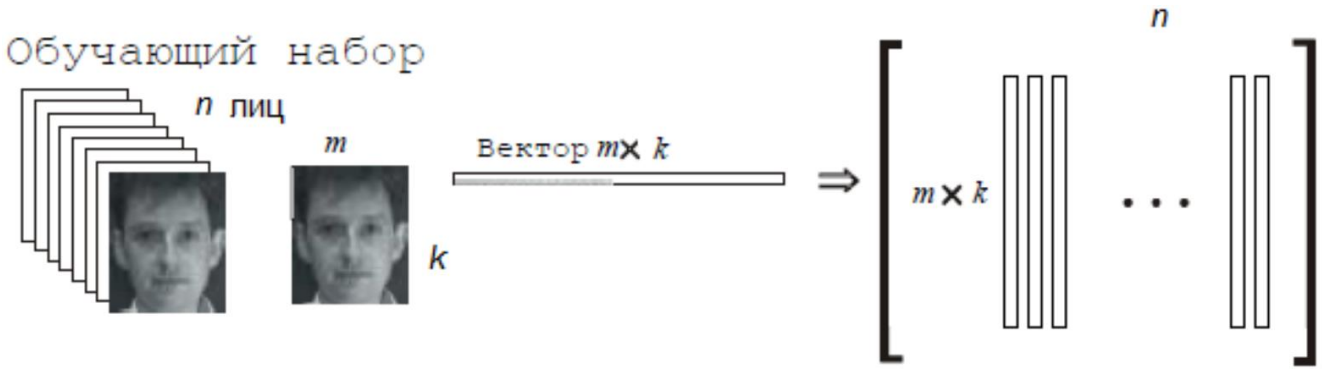

Рисунок 1.21 - Перетворення навчального набору облич на одну загальну матрицю X

Після цього проводиться нормування даних та приведення рядків до 0-го середнього та 1-ї дисперсії, обчислюється матриця коваріації. Для отриманої матриці коваріації вирішується завдання визначення власних значень та відповідних їм власних векторів (власних осіб). Далі проводиться сортування власних векторів у порядку зменшення своїх значень і залишають тільки перші k векторів за правилом:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} > (0.9 \text{ or } 0.95)$$

Отримана метрика зіставляється на графіках з еталонами бази даних, як зображено на рисунку 1.22.

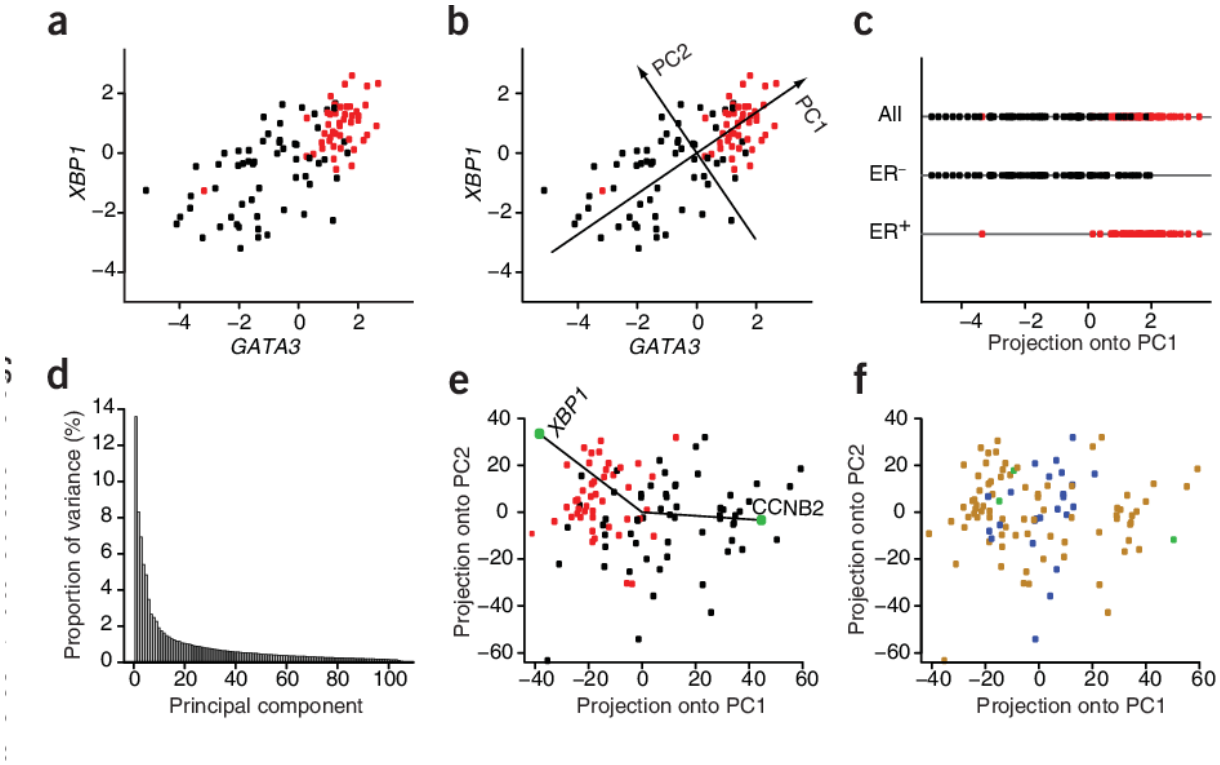

Рисунок 1.22 – Зіставлення отриманних результатів обчислень з еталонами БД

Недоліки: метод головних компонент добре зарекомендував себе у практичних додатках. Однак, у тих випадках, коли на зображенні особи присутні значні зміни у освітленості або виразі обличчя, ефективність методу значно зменшується. Справа в тому, що PCA вибирає підпростір з такою метою, щоб максимально апроксимувати вхідний набір даних, а не виконати дискримінацію між класами осіб.

1.4 Порівняння методів детектування обличчя

Практично у всіх алгоритмах обов'язковим етапом, що передує класифікації, є вирівнювання, під яким розуміється вирівнювання зображення обличчя у фронтальне положення щодо камери або приведення сукупності осіб (наприклад,

навчальній вибірці для навчання класифікатора) до єдиної системи координат. Для цього етапу необхідна локалізація на зображенні характерних всім осіб антропометричних точок – найчастіше це центри зіниць чи куточки очей. Різні дослідники виділяють різні групи таких точок. З метою скорочення обчислювальних витрат для систем реального часу розробники виділяють трохи більше 10 таких точок.

З метою оцінки ефективності запропонованих алгоритмів розпізнавання обличчя, агентство DARPA та дослідницька лабораторія армії США розробили програму FERET (face recognition technology).

У тестах програми FERET брали участь алгоритми, що ґрунтуються на гнучкому порівнянні на графах та всілякі модифікації методу головних компонентів (PCA). Ефективність всіх алгоритмів приблизно однакова. У цьому практично неможливо провести чітку різницю між ними. Для фронтальних зображень прийнятна точність розпізнавання, як правило, становить 95%. Для зображень, зроблених різними апаратами та при різному освітленні, точність, як правило, падає до 80%. Для зображень, зроблених із різницею на рік, точність розпізнавання становила приблизно 50%. Варто зауважити, що 50 відсотків — це більш ніж прийнята точність роботи подібної системи.

1.5 Висновки до розділу

Проаналізовано існуючі системи контролю втоми водія. Сформульована низка проблем для удосконалення методів детектування обличчя водія та розпізнавання стану його очей. Основною відмінністю всіх представлених алгоритмів є обчислення ознак та порівняння їх сукупностей між собою. Можна зробити висновок, що кожен з пропонованих алгоритмів використовує базу даних з великою кількістю інформації для зіставлення ознак обличчя, це впливає на кожен з алгоритмів і зменшує швидкість його роботи.

Необхідно покращити чутливість комп’ютерного зору для тимчасового спостереження інтервалу сплющення очей водія більше аніж 2 секунди за допомогою методів комп’ютерного зору бібліотеки Dlib.

Існуючі системи в разі попередження водія про критичну втомленість сповіщають водія спеціальнім звуком, яким можна знехтувати із-за людського фактора. Потрібно забезпечити не тільки звуковий сигнал, а і тимчасові кооперативні сигнали направлені на вібрацію крісла водія та зменшення обертів двигуна;

Удосконалити метод розпізнавання очей водія, забезпечивши систему інфрачервоною камерою в системі. Камера є незалежною від зовнішніх факторів таких як: ніч, погане освітлення, зв’язок з Інтернетом або з вишкам мобільного зв’язку. Детектування очей водія буде незалежне від його положення або наявності сонцезахисних окулярів.

2 ДОСЛІДЖЕННЯ УДОСКОНАЛЕННЯ МЕТОДУ СПОСТЕРЕЖЕННЯ ЗА ОЧИМА ВОДІЯ

Використання бібліотек мови програмування Python дозволяє виконати програмну реалізацію системи виявлення сонливості водія, що дозволяє визначати, як довго у конкретної людини (водія) були заплющені очі[9]. Якщо очі були закриті протягом певного часу, слід припустити, що водій починає засинати, і включити звуковий сигнал, щоб розбудити водія і привернути його увагу.

Для успішного розпізнавання необхідно розташувати камеру в машині, щоб можна було легко визначити особу водія в той момент, коли він знаходиться за кермом, та застосувати локалізацію особових ознак для спостереження за очима.

2.1 Аналіз методів детектування стану обличчя бібліотеки dlib

Класифікація стану очей в пропонованому удосконаленню існуючих алгоритмів здійснюється за допомогою методів комп'ютерного зору бібліотеки мови програмування Python – «dlib».

Dlib — багатоплатформова бібліотека загального призначення, написана на мові C++. Вона була розроблена під суттєвим впливом ідей проектування за контрактом та компонентно-орієнтованого програмування. Таким чином, вона є, перш за все, набором незалежних програмних компонентів. Це відкрите програмне забезпечення, яке випускається під ліцензією Boost Software. Оскільки розробка почалася ще 2002 року, Dlib містить широкий спектр інструментів. На 2016 рік вона містить програмні компоненти для роботи з комп'ютерними мережами, потоками, графічні інтерфейси користувача, структурами даних, лінійною алгеброю, машинним навчанням, обробки зображень, добуванням даних, XML та парсингу тексту, числової оптимізації, Баєсовими мережами та багато іншого . В останні роки, основний розвиток припав на створення широкого спектру статистичних

інструментів машинного навчання і в 2009 році Dlib було опубліковано в Journal of Machine Learning Research.

DLib – це бібліотека C++ з відкритим вихідним кодом, що реалізує різні алгоритми машинного навчання, включаючи класифікацію, регресію, кластеризацію, перетворення даних та структуроване прогнозування, а також безліч алгоритмів машинного навчання і підтримує такі функції, як багатопоточність та робота в мережі. DLib-ml реалізує безліч алгоритмів машинного навчання. Метод розпізнавання очей dlib наведено на рисунку 2.1.

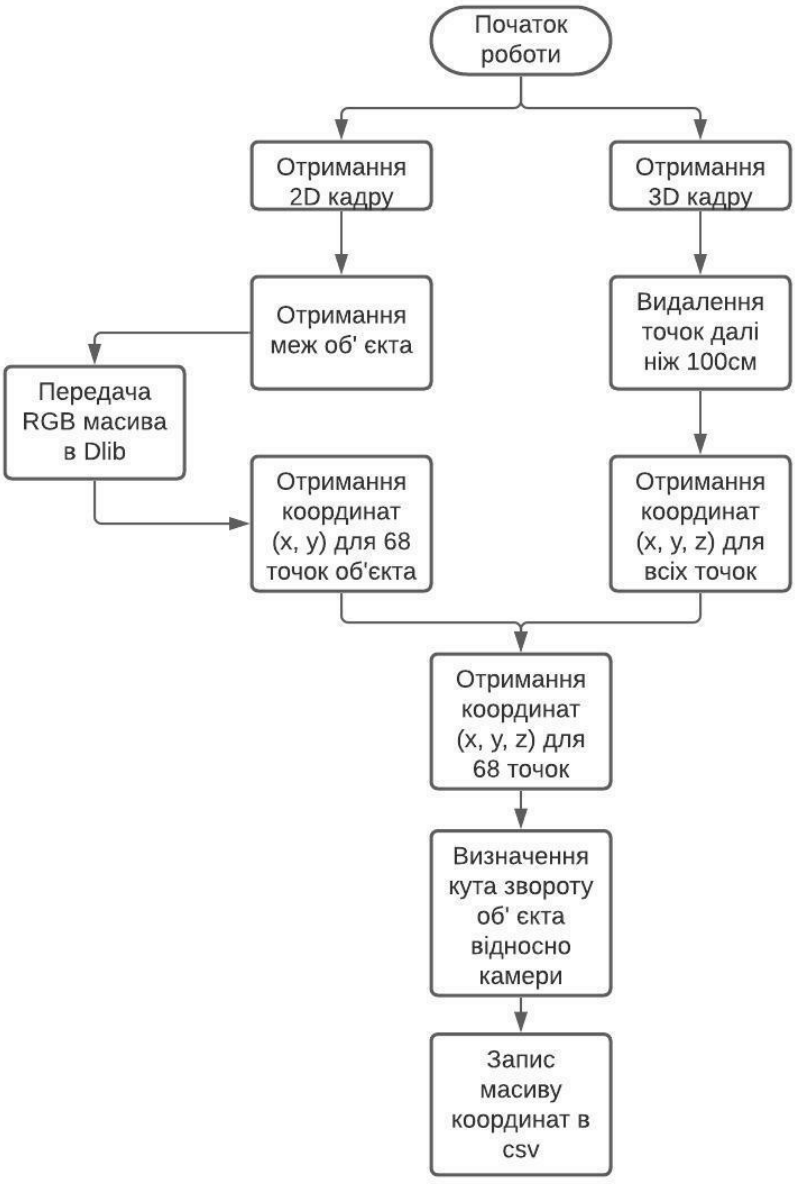

Рисунок 2.1 - Метод розпізнавання облич dlib

DLib включає широке покриття модульного тестування та приклади використання бібліотеки. Усі класи та функції бібліотеки задокументовані. Цю документацію можна знайти на домашній сторінці бібліотеки. DLib забезпечує хорошу основу для розробки програм машинного навчання на C++.

DLib дуже схожий на DMTL у тому, що він надає загальний високопродуктивний інструментарій машинного навчання з безліччю різних алгоритмів, але DLib оновлений нещодавно і містить більше прикладів. DLib також містить набагато більше допоміжних функцій.

Що робить DLib унікальним, так це те, що він призначений як для використання в дослідженнях, так і для створення програм машинного навчання на C++ та Python[10].

Бліотека Dlib являється більш точною для виявлення обличчя на зображеннях, ніж OpenCV. Dlib має більше моделей розпізнавання осіб, які можуть виявляти 68 або більше характерних точок на обличчі[11]. Ознаки (орієнтири) особи, створені dlib, є індексованим списком, який зображено на рисунку 2.2.

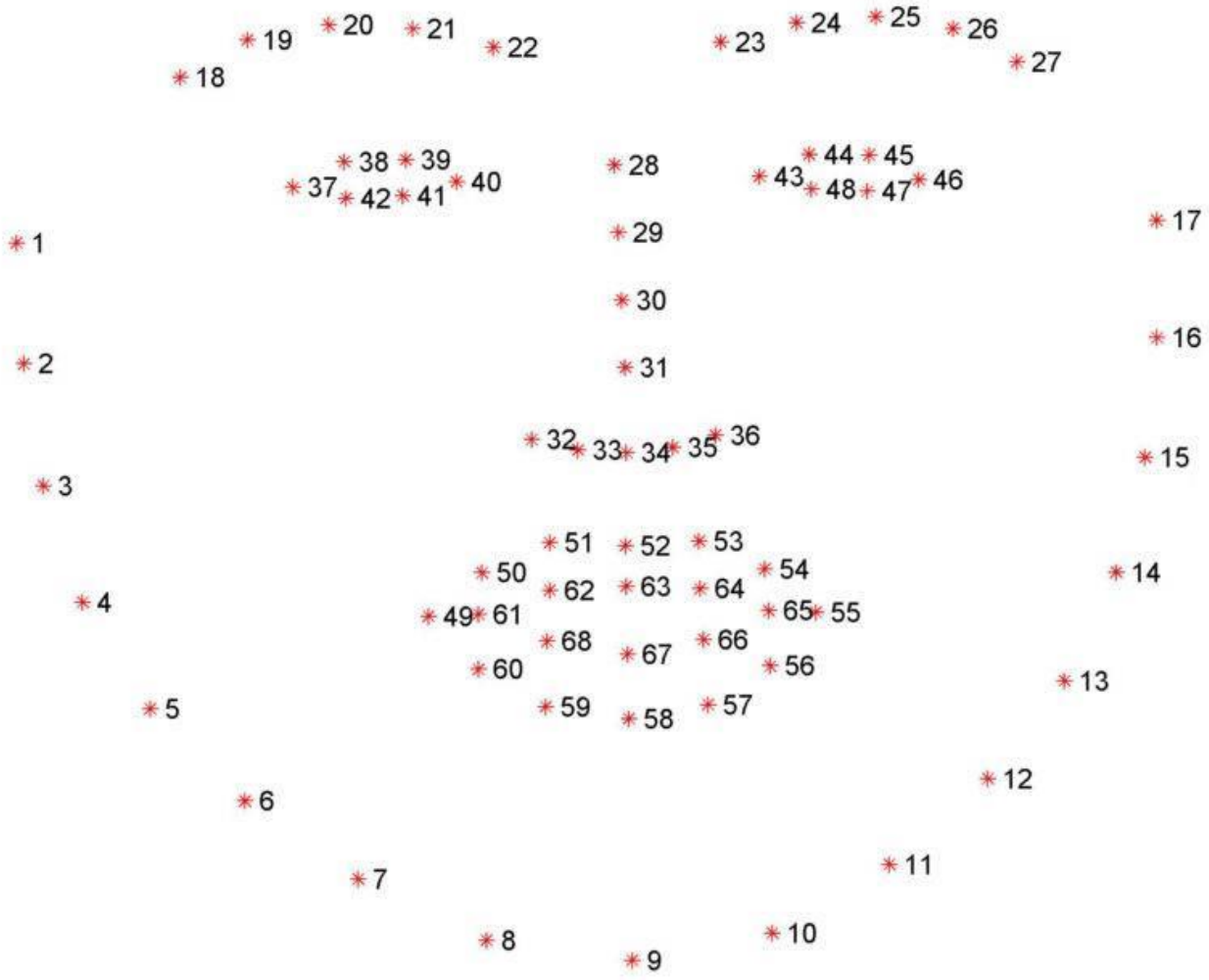

Рисунок 2.2 – Візуалізація ознак обличчя

Отже, щоб витягти області очей із набору лицьових ознак, необхідно знати правильні індекси зрізів масиву. Використовуючи ці індекси можна легко виділити області очей за допомогою спискових зрізів.

2.2 Побудова удосконаленого методу системи детектування очей водія

Спочатку створюється екземпляр потоку VideoStream, який використовує індекс інфрачервоної камери для захоплення відео. Також тут необхідно зробити паузу на одну секунду, щоб дати сенсору камери увімкнутися. У циклі програми будуть зчитуватися кадри відеопотоку один за одним, після цього вони піддаються попередньої обробки (розмір кожного кадру змінюється до ширини 450 пікселів;

кожен кадр перетворюється на відтінки сірого). Детектор облич dlib застосовується для пошуку та визначення розташування обличчя на зображенні.

Наступним кроком буде застосування функції розпізнавання ознак на обличчі, щоб локалізувати кожну важливу область обличчя. Кожне з виявлених облич перебирається в циклі (насправді передбачається, що є лише одна особа – водія, але цей цикл може бути використаний для відео з більш ніж однією особою). Для кожного з виявлених осіб застосовується детектор лицьових ознак dlib, і результат перетворюється на numpy-масив. Використовуючи зрізи масиву, можна витягти (x, y)-координати лівого та правого ока відповідно. Знаючи (x, y)-координати для обох очей, легко потім вирахувати їх співвідношення сторін. Для кращої оцінки ці значення усереднюються. Потім можна візуалізувати кожну з областей очей на кадрі за допомогою функції cv2.drawContours - це часто буває корисно, коли необхідно переконатися, що очі правильно виявляються та локалізуються.

На даному етапі, все готове для того, щоб можна було перевірити, чи водій у відеопотоці не починає проявляти симптоми сонливості.

Спочатку перевіряється, чи співвідношення сторін ока нижче порога «blink / closed» («моргання / закриття»).

Якщо це так, то збільшується лічильник, загальна кількість послідовних кадрів, у яких водій заплющив очі.

Якщо значення лічильника перевищує встановлену максимальну кількість кадрів, можна припустити, що водій починає засипати.

Далі виконується ще одна перевірка, щоб побачити, чи звуковий сигнал включений – якщо ні, то він включається.

Потім обробляється відтворення звукового сигналу. Особлива увага тут приділяється створенню окремого потоку, який відповідає за виклик звукового сигналу, щоб гарантувати, що основна програма не заблокується доти, доки не закінчиться відтворення звуку.

Після цього у кадрі відображається текст "drowsiness alert!" («попередження про сонливість»).

Нарешті, обробляється випадок, коли співвідношення сторін очей більше, ніж значення порога, що свідчить про те, що очі відкриті. Якщо очі відкриті, лічильник скидається, а звукове повідомлення не відтворюється.

Співвідношення сторін очей, а саме обчислення відношення відстаней між вертикальними ознаками ока та відстанями між горизонтальними ознаками ока, знаходяться завдяки обчисленню Евклідових відстаней між двома наборами сторін ока.

Евклідова відстань є геометричною відстанню у багатовимірному просторі.

Евклідова відстань між точками x та y у n-мірному просторі обчислюється за такою формулою[12]:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Евклідова відстань (і його квадрат) обчислюється за вихідними, а не за стандартизованими даними.

Це дає можливість використовувати незалежні змінні (у регресійному рівнянні), як про визначальний багатовимірний простір, в якому можна побудувати кожне спостереження.

Група ізометрії евклідового простору.

У математиці , а Евклідова група - це група (евклідових) ізометрій евклідового простору $\mathbb{E}$; тобто перетворення цього простору, які зберігають евклідову відстань між будь-якими двома точками (також звані евклідовими перетвореннями). Група залежить від розмірності n простору і зазвичай позначається E (n) або ISO (n)[13].

Евклідова група E (n) включає всі трансляції, обертання та відображення точки $\mathbb{E}$; та довільні кінцеві їх комбінації. Евклідова група може розглядатися як

група симетрії самого простору і містить групу симетрій будь-якої фігури (підмножини) цього простору.

Евклідові групи - це не тільки топологічні групи, це групи Лі, так що обчислення поняття можуть бути негайно адаптовані до цього налаштування.

Евклідова група E (n) є підгрупою афінної групи для вимірювань n і таким чином, щоб поважати напівпряма товарна структура обох груп. Це дає a fortiori два способи запису елементів у явній нотації. Це:

1) парою (A, b), де A - ортогональна матриця розміру n × n , a b - дійсний вектор-стовпець розміру n; або

2) однією квадратною матрицею розміру n + 1, як пояснено афінної групи .

У термінах Фелікса Кляйна Ерлангенської програми, ми читаємо з того, що евклідова геометрія, геометрія евклідової групи симетрій, тому є спеціалізацією афінної геометрії. Застосовуються всі афінні теореми. Походження евклідової геометрії дозволяє визначити поняття відстані, з якого потім можна вивести кут.

Ці групи є одними з найстаріших і найбільш вивчених, принаймні у випадках виміру 2 і 3 - неявно, задовго до винаходу концепції групи.

Це свідчить про те що кожен з пропонованих алгоритмів використовував базу даних з великою кількістю інформації для зіставлення ознак обличчя, що зменшало швидкодію кожного алгоритму. Проте в пропонованому удосконаленні алгоритму метод детектування очей не використовує базу даних для зіставлення ознак, алгоритм лише використовує простий індексований список із 68-ми характерних точок обличчя бібліотеки DLib та евклідове обчислення, яке перетворюється в матрицю(NymPy масив), завдяки якій швидко обчислюється поріг стану очей водія. Пропонується алгоритм для методу спостереження за очима водія, зображений на рисунку 2.3.

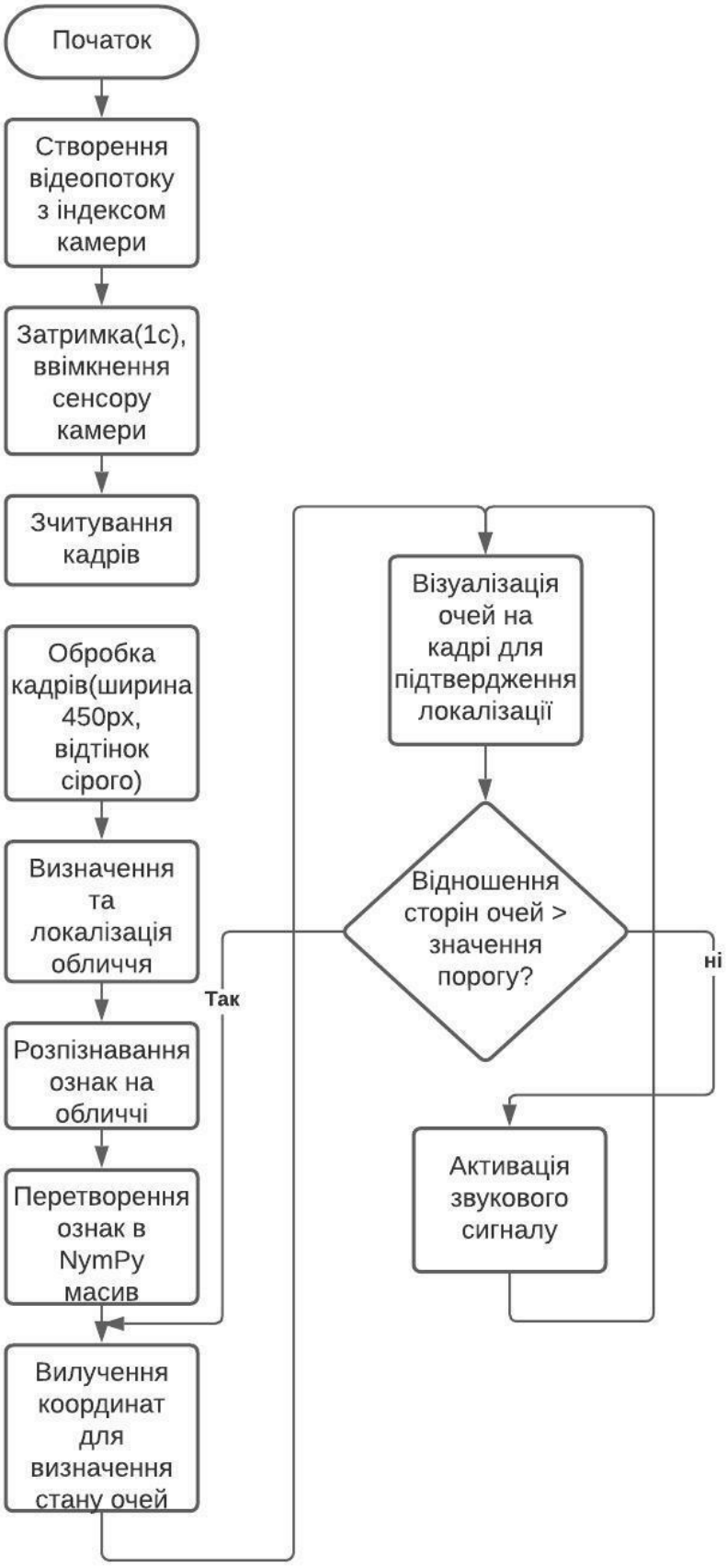

Рисунок 2.3 – Алгоритм роботи удосконаленого методу спостереження за очима

2.3 Висновки до розділу

В розділі проаналізований інструмент розпізнавання стану облич, бібліотеку DLib, завдяки якому пропоновано удосконалення існуючих методів розпізнавання та алгоритмів машинного зору. Це високопродуктивний інструментарій машинного навчання з безліччю різних алгоритмів. Так як кожен з аналізованих в попередньому розділі алгоритмів використовував базу даних з великою кількістю інформації для зіставлення ознак обличчя, що зменшало швидкодію кожного алгоритму. Пропоноване удосконалення алгоритму методу детектування очей не використовує базу даних для зіставлення ознак, алгоритм лише використовує простий індексований список із 68-ми характерних точок обличчя бібліотеки DLib та евклідове обчислення, яке перетворюється в матрицю(NymPy масив), завдяки якій швидко обчислюється поріг стану очей водія. Це дає змогу покращити швидкодію системи розпізнавання стану обличчя водія для уникнення аварійних ситуацій, дозволяючи виявити стан очей в той самий момент, коли вони заплющились.

3 МОДЕЛЮВАННЯ СИСТЕМИ. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ЗАПРОПОНОВАНОГО МЕДОДУ

Під час дорожнього руху, якщо водій почне засинати, необхідно одразу ж попередити його о можливих неприємностях, у крайній випадках - аварії. Подібне виявлення досягається за допомогою детектування стану очей водія.

Використання бібліотек мови програмування Python дозволяє виконати програмну реалізацію системи виявлення сонливості водія, що дозволяє визначати, як довго у конкретної людини(водія) були заплющені очі. Якщо очі були закриті протягом певного часу, слід припустити, що водій починає засинати, і включити звуковий сигнал, щоб розбудити водія і привернути його увагу.

Для успішного розпізнавання необхідно розташувати інфрачервону камеру в машині, щоб можна було легко визначити обличчя водія в той момент, коли він знаходиться за кермом, та застосувати локалізацію ознак для спостереження за очима. Навіть при наявності поганого освітлення завдяки функціям спеціальної камери, обличчя водія буде постійно та безперервно чітко локалізуватися і детектування очей не буде проблемою.

Класифікація стану очей здійснюється з допомогою методів комп'ютерного зору.

3.1 Опис необхідних інструментів та обладнання для моделювання системи

Пропонована система буде мати інфрачервону камеру, яка буде розташована на верхньому кордоні лобового скла машини або на торпеді перед водієм. Це дозволить камері зостережитися конкретно на водієві і не випускати його з об’єктиву. Система буде безпомилково працювати навіть при поганому освітленні обличчя водія, або в нічний час. Сонцезахисні окуляри на водієві також не стануть

бар’єром для роботи системи. Систему буде реалізовано на мові програмування Python з підключенням бібліотек: Dlib, NumPy, OpenCV.

3.1.1 Вибір камери для пропонованої системи

Камери з інфрачервоним підсвічуванням оснащені світлодіодами, що передають інфрачервоне випромінювання, практично невидиме людському оку. Чутлива матриця камери відеоспостереження розпізнає таке випромінювання, і завдяки цьому може знімати навіть у темряві. Зображення виходить монохромним і таким чітким, як за наявності освітлення. Тому система буде завжди детектувати стан очей водія

Буде встановлено інфрачервону мінікамеру «Mini cctv camera model 503», яка зображена на рисунку 3.1, з встроєним інфрачервоним підсвічуванням та звукосповіщенням.

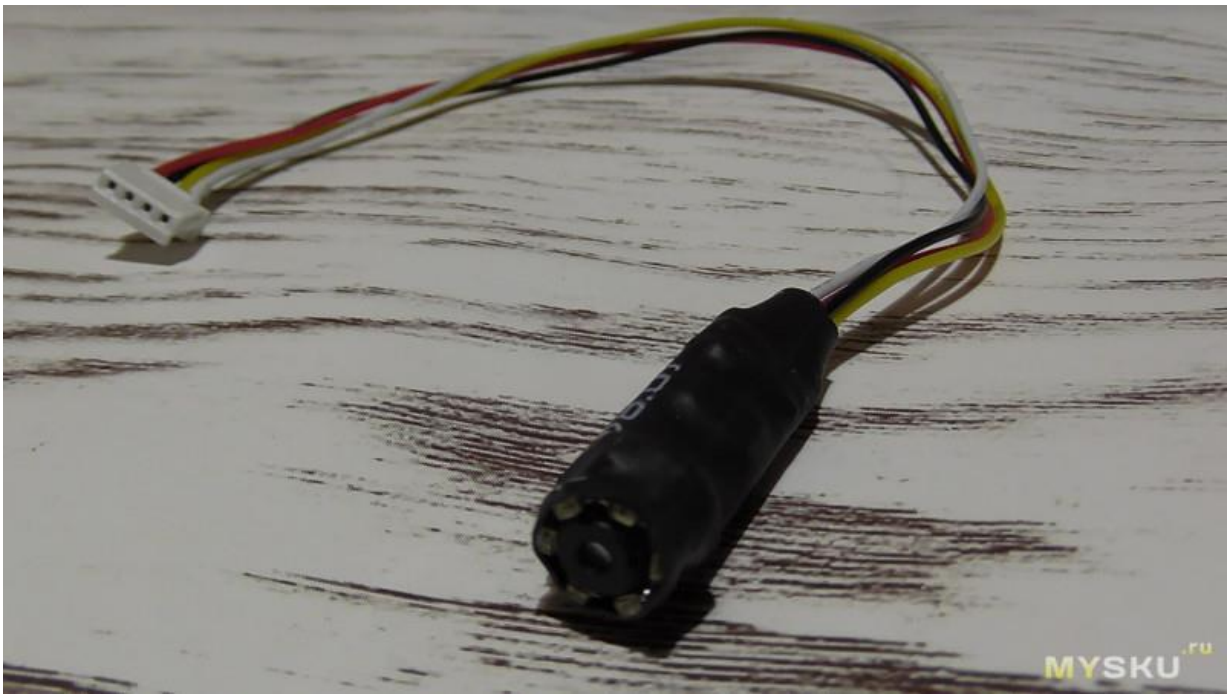

Рисунок 3.1 - Mini cctv camera model 503 з інфрачервоною підсвіткою

Характеристики:

- Датчик зображення - 1/4 PH3289;

- Розширення – 640*480px;
- Чіткість – 480TVL;
- Освітлення – 1LUX;
- Гучність звукового сигналу – 48dB;
- Гамма – 0,45;
- Зчитування кадрів – 30f/s;
- Баланс білого – auto;
- Відео вихід – CVBS 1Vp-p;
- TV формат – PAL/NTSC
- Живлення – 3-5V

Роз'єм підключення має чотири контакти:

- чорний провід - "мінус" (загальний).
- червоний провід - "плюс";
- жовтий кабель — вихід відеосигналу з камери.
- білий провід - вихід аудіосигналу

Відео та аудіо вихід підключається з чорним дротом, чорний загальний провід.

3.1.2 Мова програмування Python. Опис необхідних бібліотек

Система реалізована на мові програмуванні Python, рисунок 3.2. Python — високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним управлінням пам'яттю, орієнтована підвищення продуктивності розробника, читаності коду та її якості, і навіть забезпечення переносимості написаних у ньому програм. Мова є повністю об’єктно-орієнтованною — все є об’єктами. Незвичайною особливістю мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови мінімалістичний, рахунок чого практично рідко виникає необхідність звертатися до документації. Сама ж мова відома як інтерпретована і використовується в тому числі для написання скриптів. Недоліками мови є найчастіше нижча швидкість роботи та

більш високе споживання пам'яті написаних на ньому програм порівняно з аналогічним кодом, написаним компілюваними мовами, таких як C або C++ [14].

Рисунок 3.2 – Мова програмування Python

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об' єктно-орієнтоване програмування, метапрограмування та функціональне програмування. Завдання узагальненого програмування вирішуються рахунок динамічної типізації. Аспектно-орієнтоване програмування частково підтримується через декоратори, більш повноцінна підтримка забезпечується додатковими фреймворками. Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень. Основні архітектурні риси — динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопотокових обчислень із глобальним блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які можуть об'єднуватися в пакети.

Стандартна бібліотека включає великий набір корисних функцій, що переносяться, починаючи від функціоналу для роботи з текстом і закінчуючи засобами для написання мережевих додатків. Додаткові можливості, такі як математичне моделювання, робота з обладнанням, написання веб-додатків або

розробка ігор можуть реалізовуватися за допомогою великої кількості сторонніх бібліотек, а також інтеграцією бібліотек, написаних на C або C++, при цьому і сам інтерпретатор Python може інтегруватися в проекти, написані цими языками. Існує і спеціалізований репозиторій програмного забезпечення, написаного на Python - PyPI. Цей репозиторій надає засоби для простої установки пакетів в операційну систему та став стандартом де-факто для Python. Станом на 2019 рік у ньому містилося понад 175 тисяч пакетів.

Python став однією з найпопулярніших мов, він використовується в аналізі даних, машинному навчанні, DevOps та веб-розробці, а також в інших сферах, включаючи розробку ігор. За рахунок читабельності, простого синтаксису та відсутності необхідності в компіляції мова добре підходить для навчання програмування, дозволяючи концентруватися на вивченні алгоритмів, концептів та парадигм. Налагодження ж і експериментування значною мірою полегшуються тим фактом, що мова інтерпретується. Використовується мова багатьма великими компаніями, такими як Google або Facebook. Станом на жовтень 2021 року Python посідає перше місце у рейтингу TIOBE популярності мов програмування з показником 11,27%.

Python портований і працює майже на всіх відомих платформах від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично під всі варіанти UNIX (включаючи FreeBSD та Linux), Android, Plan 9, Mac OS та macOS, iPhone OS (iOS) 2.0 і вище, iPadOS, Palm OS, OS/2, Amiga , HaikuOS, AS/400, OS/390, Windows Mobile та Symbian.

Python підтримує динамічну типізацію, тобто тип змінної визначається лише під час виконання. Тому замість «надання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». До примітивних типів у Python відносяться булевий, ціле число довільної точності, число з плаваючою комою та комплексне число. З контейнерних типів Python вбудовані: рядок, список, кортеж, словник і безліч. Усі значення є об'єктами, зокрема функції, методи, модулі, класи.

Для відтворення звукового сигналу в системі, використано бібліотеку «playsound». Це чистий Python, кроссплатформенний, однофункціональний модуль

без залежностей для відтворення звуків. Модуль playsound містить лише одне - функцію (також звану) playsound. Потрібний один аргумент - шлях до файлу зі звуком, який потрібно відтворити. Це може бути локальний файл або URL-адреса[15].

Існує необов'язковий другий аргумент block, для якого встановлено значення True. Якщо встановлено значення False, функція запускається асинхронно.

У Windows використовується windll.winmm. WAVE та MP3 були протестовані і, як відомо, працюють. Інші формати файлів можуть працювати. OS X використовує AppKit.NSSound. Загалом все, що QuickTime може відтворювати, для OS X повинен відтворювати звук.

У Linux використовує GStreamer. Відомо, що працює з Ubuntu 14.04 та ElementaryOS Loki.

Dlib – це сучасний набір інструментів C++, що містить алгоритми машинного навчання та інструменти для створення складного програмного забезпечення на C++ для вирішення реальних проблем. Він використовується як у промисловості, так і в академічних колах у широкому діапазоні областей, включаючи робототехніку, вбудовані пристрої, мобільні телефони та великі високопродуктивні обчислювальні середовища. Ліцензування відкритого вихідного коду Dlib дозволяє вам використовувати його в будь-якій програмі безкоштовно[16].

На відміну від багатьох проектів з відкритим вихідним кодом, цей надає повну та точну документацію для кожного класу та функції. Існують також режими налагодження, в яких перевіряються задокументовані умови для функцій. Коли цей параметр увімкнено, він буде виявляти переважну більшість помилок, викликаних неправильним викликом функцій або неправильним використанням об'єктів.

Покриття модульним тестом. Відношення рядків коду модульного тесту до рядків коду бібліотеки становить приблизно від 1 до 4. Бібліотека регулярно тестується в системах MS Windows, Linux та Mac OS X. Однак він повинен працювати в будь-якій системі POSIX та використовувався у Solaris, HPUX та BSD.

Немає жодних інших пакетів для використання бібліотеки. Потрібні лише ті API-інтерфейси, що надаються готовою ОС. Перед використанням бібліотеки не потрібні жодні кроки встановлення або налаштування.

Весь код, специфічний для операційної системи, ізольований усередині рівнів абстракції ОС, збережених якнайменше. Решта бібліотеки або розташована поверх рівнів абстракції ОС, або є чистим стандартом C++ ISO.

Для розрахунків евклідової відстані, перетвореня результатів розрахунків в матрицю та отримання певних потрібних метрик, потрібно імпортувати бібліотеку NumPy, рисунок 3.3.

Рисунок 3.3 – Можливості бібліотеки NumPy

Це забезпечує:

- потужний об'єкт N-мірного масиву
- складні (широкомовні) функції
- інструменти для інтеграції коду C/C++ та Fortran
- корисні можливості лінійної алгебри, перетворення Фур'є та випадкових чисел і багато іншого.

Крім очевидного наукового використання, NumPy можна також використовувати як ефективний багатовимірний контейнер загальних даних.

Можуть бути визначені типи даних. Це дозволяє NumPy легко та швидко інтегруватися з широким спектром баз даних. Усі колеса NumPy, що розповсюджуються на PyPI, мають ліцензію BSD[17].

Для написання, компілювання і тестування програми системи обрано PyCharm. PyCharm - IDE для професійної розробки на Python[18]. Максимальна продуктивність. PyCharm подбає про рутинні завдання, під час яких можливо зосередитися на більш важливих речах. Працюючи в PyCharm, заощаджується час.

Розумний механізм аналізу коду забезпечує точне автодоповнення, пошук помилок та швидкі виправлення, зручну навігацію за кодом та інші корисні функції.

PyCharm допомагає писати код, який легко підтримувати. IDE контролює якість коду за допомогою перевірок відповідності вимогам PEP8, розумних рефакторингів та безлічі інспекцій, а також надає допомогу при тестуванні. PyCharm створюється програмістами для програмістів, тому в ньому є все потрібне для продуктивної розробки на Python.

OpenCV (бібліотека комп'ютерного зору з відкритим вихідним кодом: http://opencv.org) - це бібліотека з відкритим вихідним кодом, що включає кілька сотень алгоритмів комп'ютерного зору, рисунок 3.4. У документі описується так званий API OpenCV 2.x, який, по суті, є API C++, на відміну від API OpenCV 1.x на основі C (API C застарів і не тестується з компілятором C, починаючи з випусків OpenCV 2.4)[19].

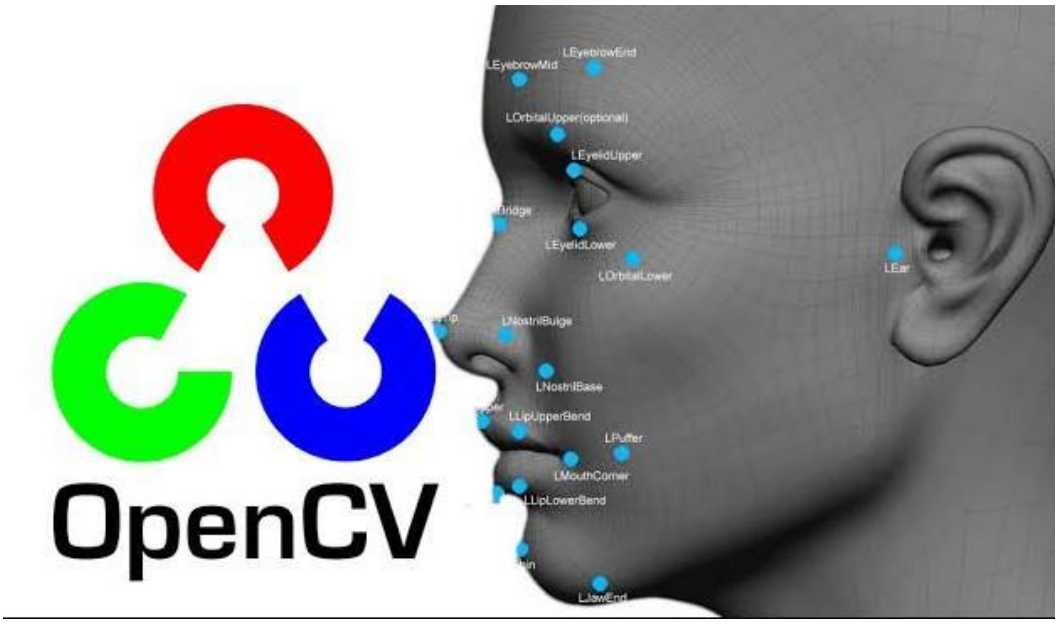

Рисунок 3.4 - Бібліотека комп'ютерного зору OpenCV

OpenCV має модульну структуру, що означає, що пакет включає кілька загальних чи статичних бібліотек. Доступні такі модулі:

1) Базова функціональність (core) - компактний модуль, що визначає базові структури даних, включаючи щільний багатовимірний масив Mat і базові функції, що використовуються іншими модулями.

2) Обробка зображень (imgproc) - модуль обробки зображень, який включає лінійну і нелінійну фільтрацію зображень, геометричні перетворення зображень (зміна розміру, афінне і перспективне спотворення, універсальне перетворення на основі таблиць), перетворення колірного простору, гістограми і т.д.

3) Відеоаналіз (відео) - модуль відеоаналізу, який включає алгоритми оцінки руху, віднімання фону і відстеження об'єктів.

4) Калібрування камери та 3D-реконструкція (calib3d) - базові алгоритми множинної геометрії, калібрування одиночної та стереокамери, оцінка пози об'єкта, алгоритми стереовідповідності та елементи 3D-реконструкції.

5) 2D Features Framework (features2d) - детектори основних функцій, дескриптори та співставники дескрипторів.

6) Виявлення об'єктів (objdetect) - виявлення об'єктів та екземплярів зумовлених класів (наприклад, осіб, очей, кухлів, людей, автомобілів тощо).

7) Високоврівневий графічний інтерфейс (highgui) - простий у використанні інтерфейс для простих можливостей інтерфейсу користувача.

8) Відео введення / виведення (videoio) - простий у використанні інтерфейс для захоплення відео та відеокодеків.

9) Деякі інші допоміжні модулі, такі як тестові оболонки FLANN і Google, Python прив'язки та інші.

3.2 Побудова алгоритму програми. Практична реалізація системи

Спочатку створюється екземпляр потоку VideoStream, який використовує індекс веб-камери ноутбука для захоплення відео. Також тут необхідно зробити паузу на одну секунду, щоб дати сенсору камери увімкнутися. У циклі зчитуються кадри відеопотоку один за одним, і потім вони піддаються попередньої обробки (розмір кожного кадру змінюється до ширини 450 пікселів; кожен кадр перетворюється на відтінки сірого). Детектор облич dlib застосовується для пошуку та визначення розташування особи (осіб) на зображенні.

Наступним кроком буде застосування функції розпізнавання ознак на обличчі, щоб локалізувати кожну важливу область обличчя. Кожна з виявлених осіб перебирається в циклі (насправді передбачається, що є лише одна особа – водія, але цей цикл може бути використаний для відео з більш ніж однією особою). Для кожного з виявлених осіб застосовується детектор лицьових ознак dlib, і результат перетворюється на numpy-масив. Використовуючи зрізи масиву, можна витягти (x, y)-координати лівого та правого ока відповідно. Знаючи (x, y)-координати для обох очей, легко потім вирахувати їх співвідношення сторін. Для кращої оцінки ці значення усереднюються. Потім можна візуалізувати кожну з областей очей на кадрі за допомогою функції cv2.drawContours - це часто буває корисно, коли необхідно переконатися, що очі правильно виявляються та локалізуються. Блок-схему алгоритму програми зображено на рисунку 3.5.

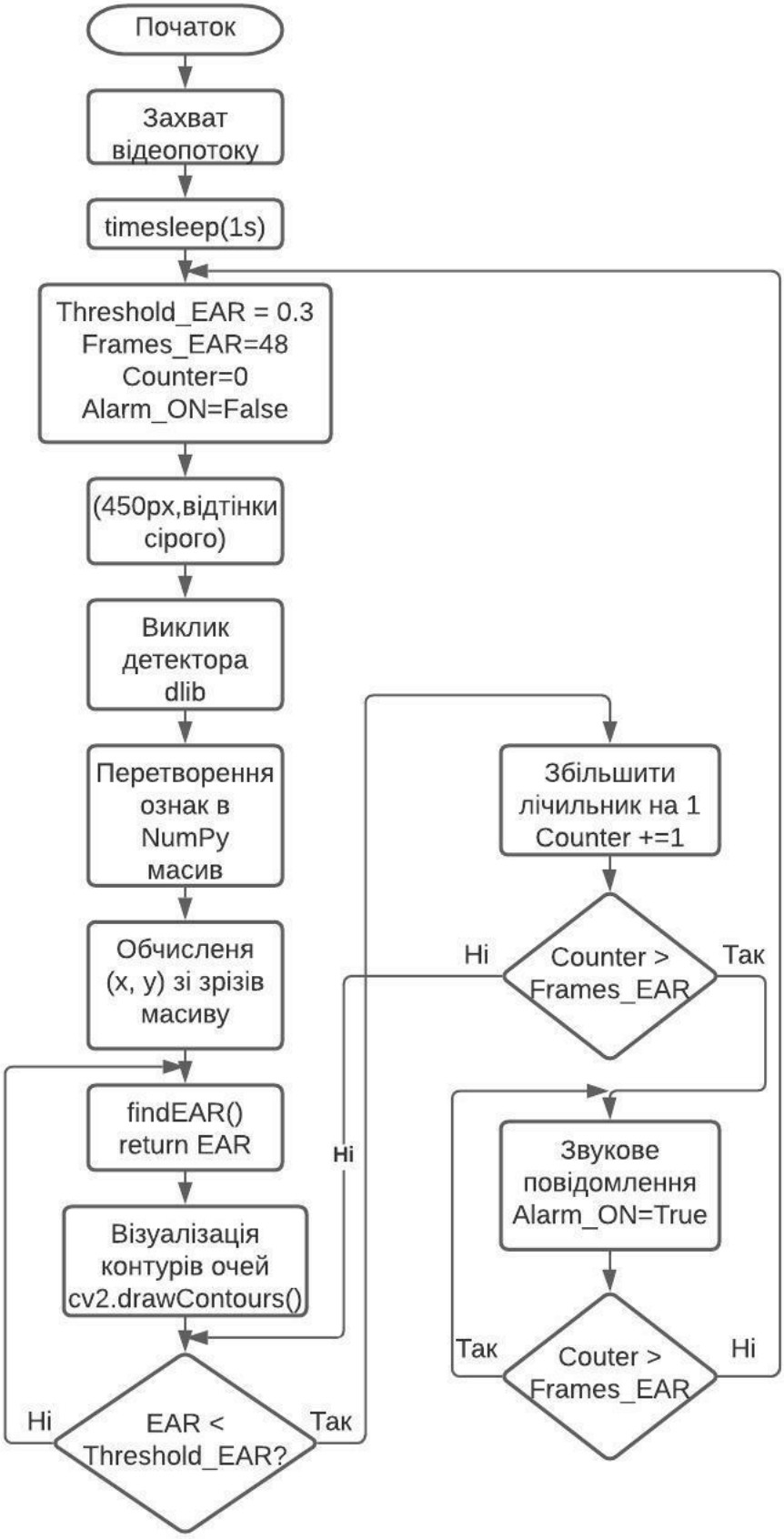

Рисунок 3.5 - Блок-схему алгоритму програми

Щоб розпочати реалізацію, необхідно створити новий *.py-файл, відкрити його в текстовому редакторі або середовищі розробки мови Python, в даному випадку професійний IDE – PyCharm, і здійснити підключення необхідних бібліотек. Повний лістинг програми наведено у додатку А. Вихідний код імпортованих бібліотек наведено у лістингу 1.

Лістинг 1. Імпорт бібліотек для реалізації системи:

```
import dlib
import numpy
from cv2 import videostream
from playsound import playsound
import time
```

Для відтворення звукового сигналу в системі необхідно створити звуковий поток, вказавши йому шлях до звукового файлу. Для цього використано бібліотеку playsound. Програмну функцію відтворення звуку наведено у лістингу 2.

Лістинг 2. Функція відтворення звукового повідомлення водію:

```
def runAlarm(path):
    # відтворити повідомлення
    playsound.playsound(C:\Users\asus\Music\Alarm)
```

Для обчислення відношення евклідової відстані між двома наборами горизонтальних і вертикальних ознак очей(з координатами x, y), створено функцію findEAR(). Завдяки матриці евклідової групи повертається метрика стану кожного ока. Програмну функцію обчислення евклідової відстані наведено у лістингу 3.

Лістинг 3. Функція обчислення евклідової відстані:

```
def findEAR(eye):
```

```
        # обчислити евклідові відстані між двома наборами вертикальних ознак ока (x, y)-
координат
            A = dist.euclidean(eye[1], eye[5])
            B = dist.euclidean(eye[2], eye[4])
             # те саме для горизонтальних ознак
            C = dist.euclidean(eye[0], eye[3])
            # обчислити EAR
            ear = (A + B) / (2.0 * C)
           # повернути його
             return ear
```

Значення співвідношення сторін ока, що повертається, буде приблизно константним і позитивним, коли око відкрите. Потім значення швидко зменшуватиметься до нуля під час моргання[9].

Якщо око закрите, співвідношення сторін ока знову залишиться приблизно постійним, але буде набагато менше, ніж співвідношення в той момент, коли око відкрите. Візуалізація орієнтирів очей наведена рисунку 3.6.

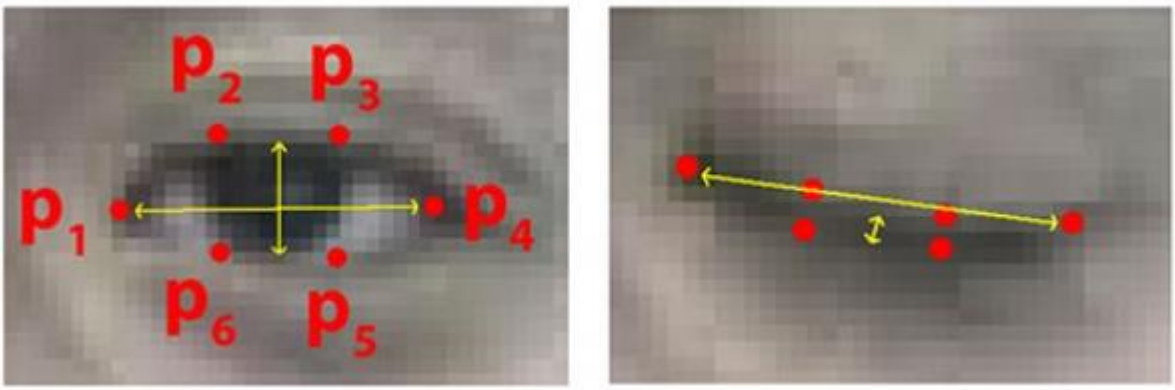

Рисунок 3.6 - Візуалізація орієнтирів очей водія

Вгорі ліворуч: око відкрите. Вгорі справа: око закрите. На рисунку 3.7 приведено графік співвідношення сторін із часом. Падіння співвідношення сторін ока вказує на моргання.

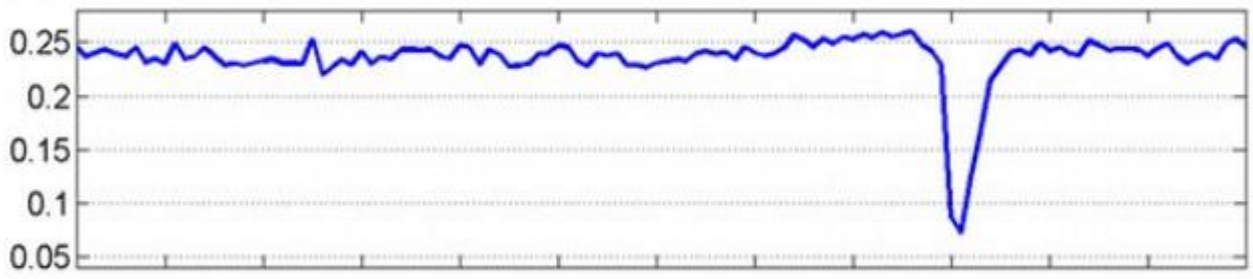

Рисунок 3.7 - Графік співвідношення сторін очей водія із часом

Для проектування детектора слід відстежувати співвідношення сторін очей, щоб побачити, чи це значення падає, але разом з тим, чи не збільшується воно знову, що означатиме, що водій закрив очі. Далі визначаються деякі важливі для подальших обчислень змінні наведені у лістингу 4.

Лістинг 4. Створення ключових змінних для детектування очей водія(порогові значення):

```
# визначити дві константи:
# одну для співвідношення сторін ока, щоб визначити моргання, а потім другу константу для кількості послідовних кадрів.
#стан ока має бути нижче порогового значення, щоб спрацював сигнал тривоги
THRESHOLD_EAR = 0.3
FRAMES_EAR = 48
# ініціалізувати лічильник кадрів, а також логічне значення, використовуване для індикації спрацьовування сигналізації
COUNTER = 0
ALARM_ON = False
```

Визначено змінну THRESHOLD_EAR. Якщо співвідношення сторін ока впаде нижче цього порога, необхідно почати підрахунок кількості кадрів, протягом яких людина заплющила очі.

Якщо кількість кадрів, у яких людина заплющила очі, перевищує FRAMES_EAR, необхідно видати звуковий сигнал.

Експериментально було виявлено, що значення THRESHOLD_EAR, що дорівнює 0,3, добре працює у різних ситуаціях.

Значення FRAMES_EAR також експериментально було встановлено на 48 – якщо людина заплющує очі на 48 послідовних кадрів, то відтворюватиметься звуковий сигнал. Можна зробити детектор чутливішим, зменшивши FRAMES_EAR – аналогічно, менш чутливим, збільшивши його.

У лістингу 3 також визначено змінну COUNTER – загальну кількість послідовних кадрів, протягом яких співвідношення сторін ока менше THRESHOLD_EAR. Якщо COUNTER перевищує FRAMES_EAR, слід оновити логічне значення ALARM_ON.

Бібліотека dlib поставляється з детектором обличчя на основі гістограми орієнтованих градієнтів разом із провісником лицьових ознак. Відповідні екземпляри створюються у наступному блоці коду наведеному у лістингу 5.

Лістинг 5. Створення екземплярів детектора обличчя і лицьових ознак водія:

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
```

Ознаки(орієнтири) обличчя, створені dlib, є список, що індексується зображеному на рисунку 3.8. Отже, щоб витягти області очей з набору ознак, необхідно знати правильні індекси зрізів масиву, як наведено у лістингу 6.

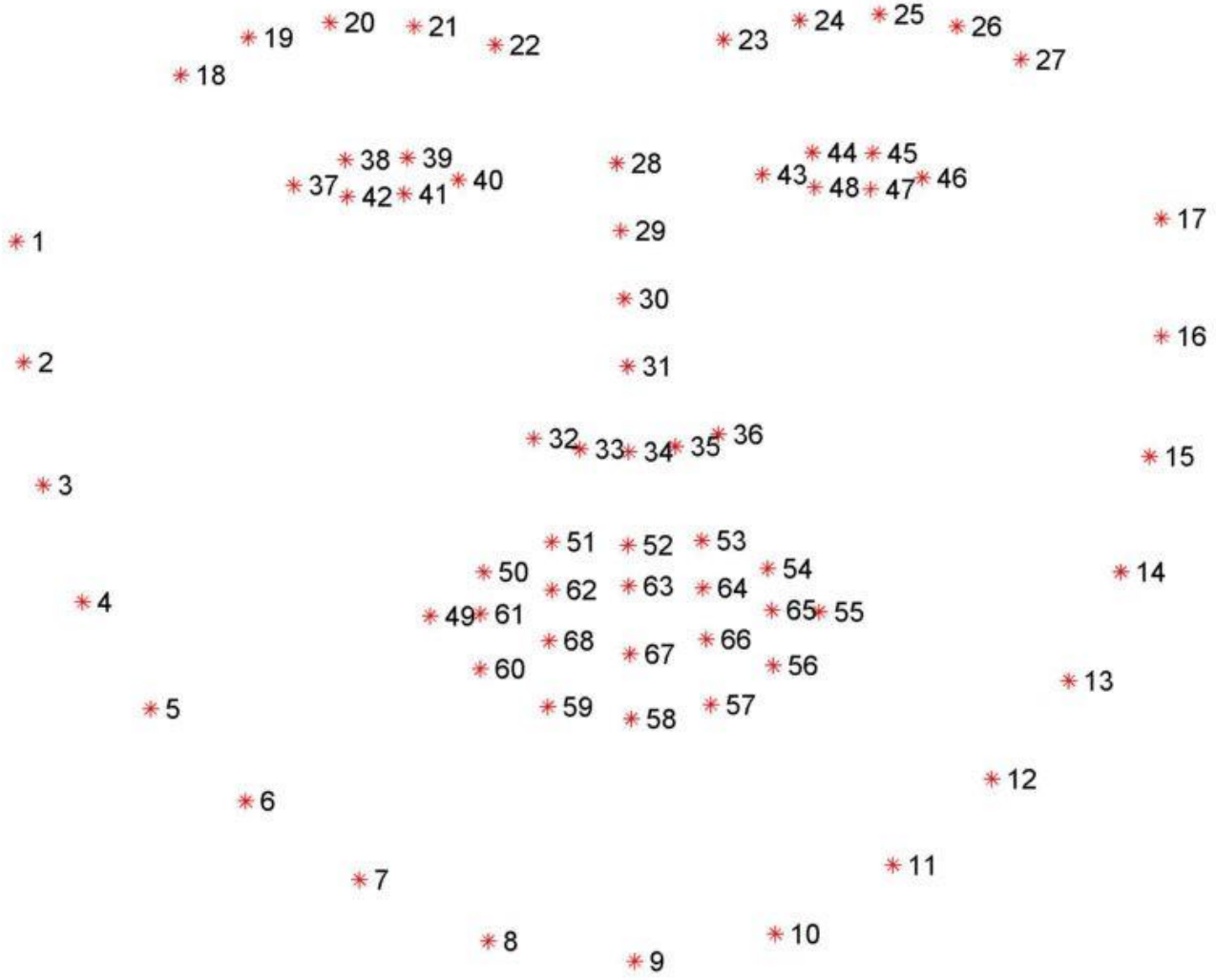

Рисунок 3.8 – Індексований список лицьових орієнтирів обличчя

Лістинг 6. Спискові зрізи індексу очей:

```
(leftEyeSt, leftEyeEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rightEyeSt, rightEyeEnd) = face_utils. FACIAL_LANDMARKS_IDXS["right_eye"]
```

Використовуючи ці індекси можна легко виділити області очей за допомогою спискових зрізів. Ядро детектора сонливості водія наведено у лістингу 7.

Лістинг 7. Ядро детектора сонливості водія:

```
# запустити захоплення відеопотоку з веб-камери
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)
```

```
цикл з кадрів з відеопотоку
while True:
# захопити кадр із потокового відеофайлу, змінити його розмір і перетворити на канали у відтінки сірого
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Виявити обличчя за допомогою детектора
    rects = detector(gray, 0)
```

Нарешті, все готове для того, щоб можна було перевірити, чи водій у відеопотоці не починає проявляти симптоми сонливості.

3.3 Експериментальна перевірка системи

Перші програмні системи розроблялися у межах програм наукових досліджень чи програм потреб міністерств оборони. Тестування таких продуктів проводилося строго формалізовано із записом всіх тестових процедур, тестових даних, отриманих результатів. Тестування виділялося в окремий процес, який починався після завершення кодування, але при цьому зазвичай виконувалося тим же персоналом[20].

Багато уваги приділяється «вичерпному» тестуванню, яке має проводитись з використанням усіх шляхів у коді або всіх можливих вхідних даних. Було зазначено, що в цих умовах повне тестування програмного забезпечення неможливе, тому що, по-перше, кількість можливих вхідних даних дуже велика, по-друге, існує безліч шляхів, по-третє, складно знайти проблеми в архітектурі та специфікаціях. З цих причин «вичерпне» тестування було відхилено та визнано теоретично неможливим. Тестування систем зображено на рисунку 3.9.

Рисунок 3.9 – Етапи тестування систем

3.3.1 Опис основних етапів тестування

Тестування програмного забезпечення (Software Testing) - перевірка відповідності між реальною та очікуваною поведінкою програми, що здійснюється на кінцевому наборі тестів, обраному певним чином. У більш широкому сенсі, тестування - це одна з технік контролю якості, що включає активності з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) та аналізу отриманих результатів (Test Analysis)[21].

Верифікація (Verification) - це процес оцінки системи або її компонентів з метою визначення, чи задовольняють результати поточного етапу розробки умов, сформованих на початку цього етапу. Тобто. чи виконуються наші цілі, терміни, завдання розробки проекту, визначені на початку поточної фази.

Валідація (Validation) – це визначення відповідності розроблюваного ПЗ очікуванням та потребам користувача, вимогам до системи.

План Тестування (Test Plan) - це документ, що описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладу, критеріїв початку та закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення .

Тест дизайн (Test Design) - це етап процесу тестування ПЗ, на якому проектуються та створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріїв якості та цілей тестування.

Тестовий випадок (Test Case) - це артефакт, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції або її частини.

Баг/Дефект Репорт (Bug Report) - це документ, що описує ситуацію або послідовність дій, що призвела до некоректної роботи об'єкта тестування, із зазначенням причин та очікуваного результату.

Тестове Покриття (Test Coverage) - це одна з метрик оцінки якості тестування, що представляє собою щільність покриття тестами вимог або виконуваного коду.

Деталізація Тест Кейсов (Test Case Specification) – це рівень деталізації опису тестових кроків та необхідного результату, при якому забезпечується розумне співвідношення часу проходження до тестового покриття

Час проходження Тест Кейса (Test Case Pass Time) – це час від початку проходження кроків тест кейсу до отримання результату тесту.

3.3.2 Покрокове тестування системи

Пропонована удосконалена система розпізнавання очей водія має інфрачервону камеру, яка буде розташована на верхньому кордоні лобового скла машини або на торпеді перед водієм, рисунок 3.10.

Рис. 3.10 – Передбачуване місце для розташування інфрачервоної камери в машині

Це дозволить камері зостережитися конкретно на водієві і не випускати його з об’єктиву. Із-за функцій інфрачервоної камери, передбачується, що система буде безпомилково працювати навіть при поганому освітленні обличчя водія, або в нічний час. Сонцезахисні окуляри на водієві також не стануть бар’єром для роботи системи. Система реалізована на мові програмування Python.

При заведенні машини, система автоматично активується. Перш за все налаштовується камера, яка відстежує всі особи на відеокадрі потоку, як показано на рисунку 3.11. Слід зауважити, що камера зосереджена конкретно на водієві, тому зайвих облич в системі розпізнано не буде.

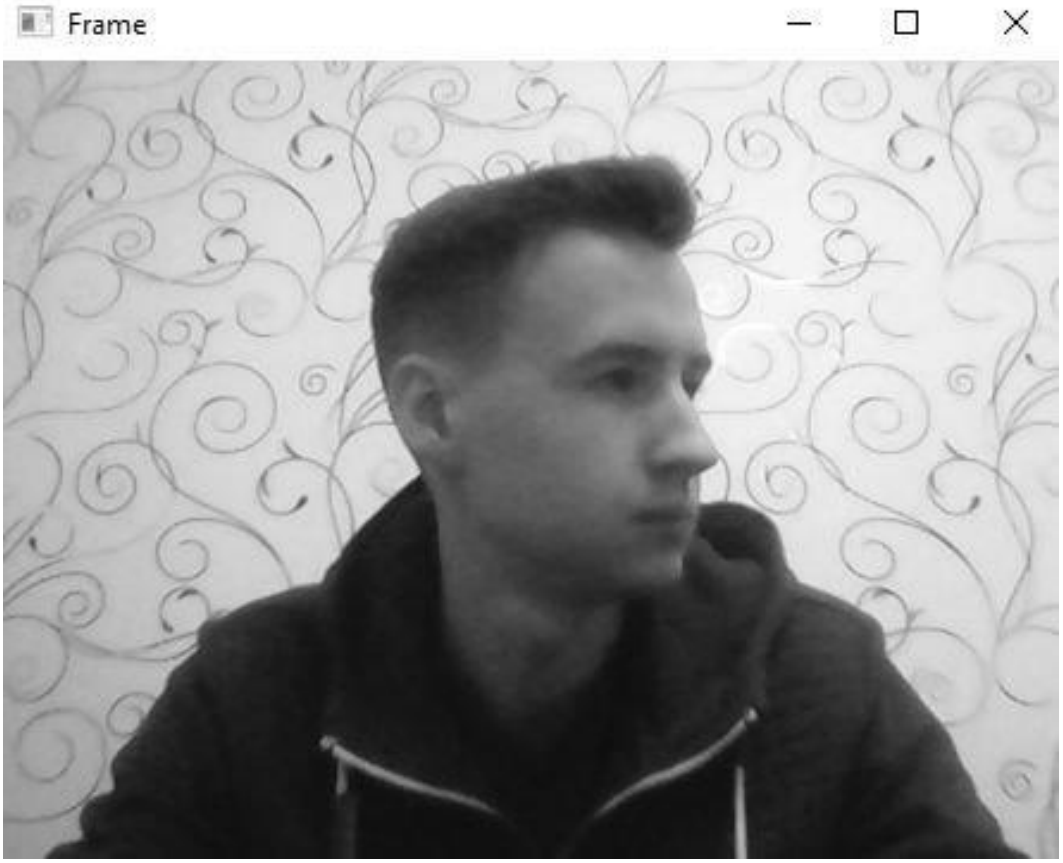

Рисунок 3.11 - Пошук облич у вхідному відеопотоці.

Коли обличчя знайдено, то виконується визначення ознак очей на обличчі та дістаються області очей, як зображено на рисунку 3.12, де EAR – поріг відношення сторін очей.

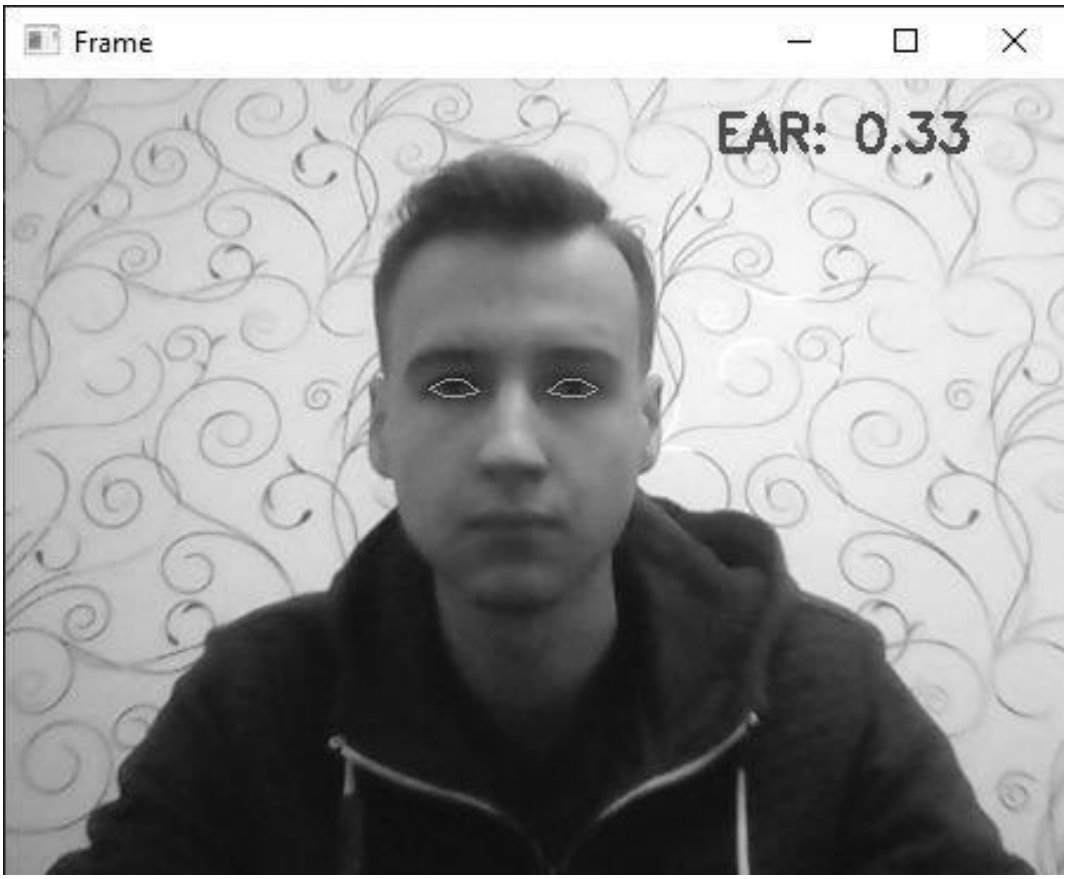

Рисунок 3.12 – Визначення ознак очей на обличчі

Написана програмна функція cv2.drawContours(), для виділення контурів очей працює без помилок. Тепер, коли на кадрі виділені області очей, можна визначити співвідношення сторін ока (EAR – Eye Aspect Ratio), щоб визначити, чи закриті очі, як зображено на рисунку 3.13.

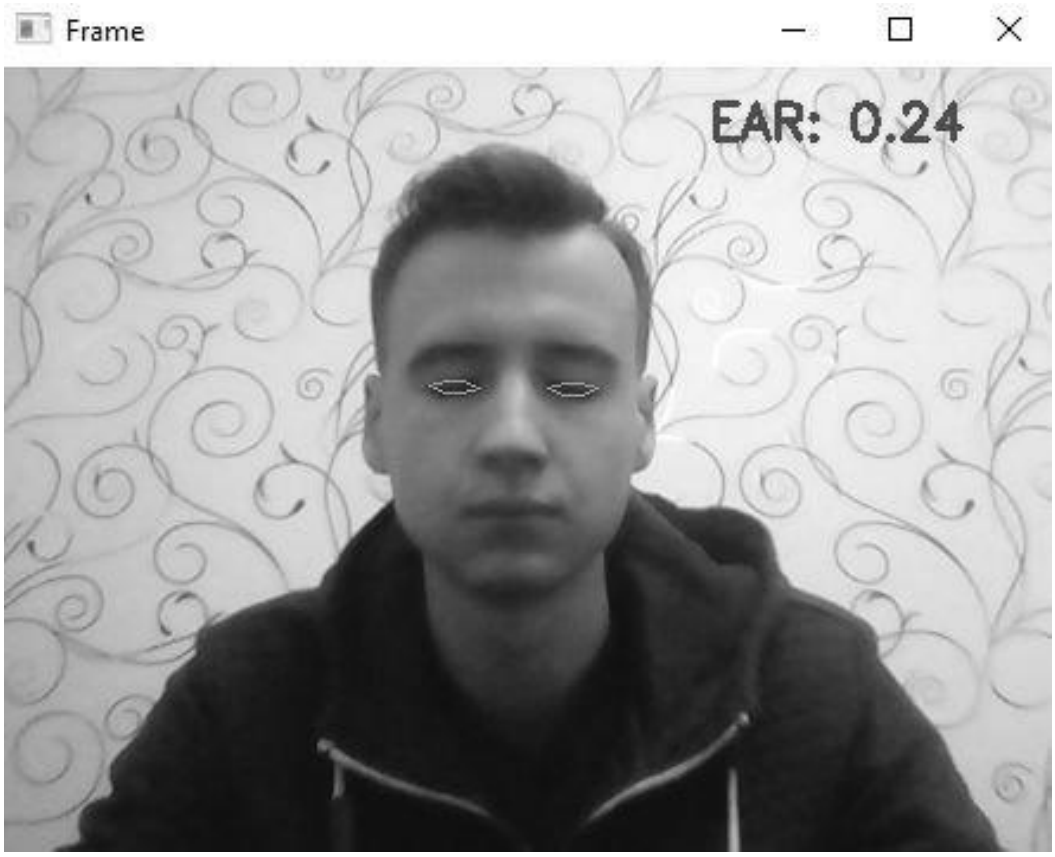

Рис. 3.13 – Визначення співвідношення сторін ока

Якщо співвідношення сторін ока вказує на те, що очі були закриті досить довгий час, то викликається звукове повідомлення, призначене для того, щоб розбудити водія, що заснув, це зображено на рисунку 3.14.

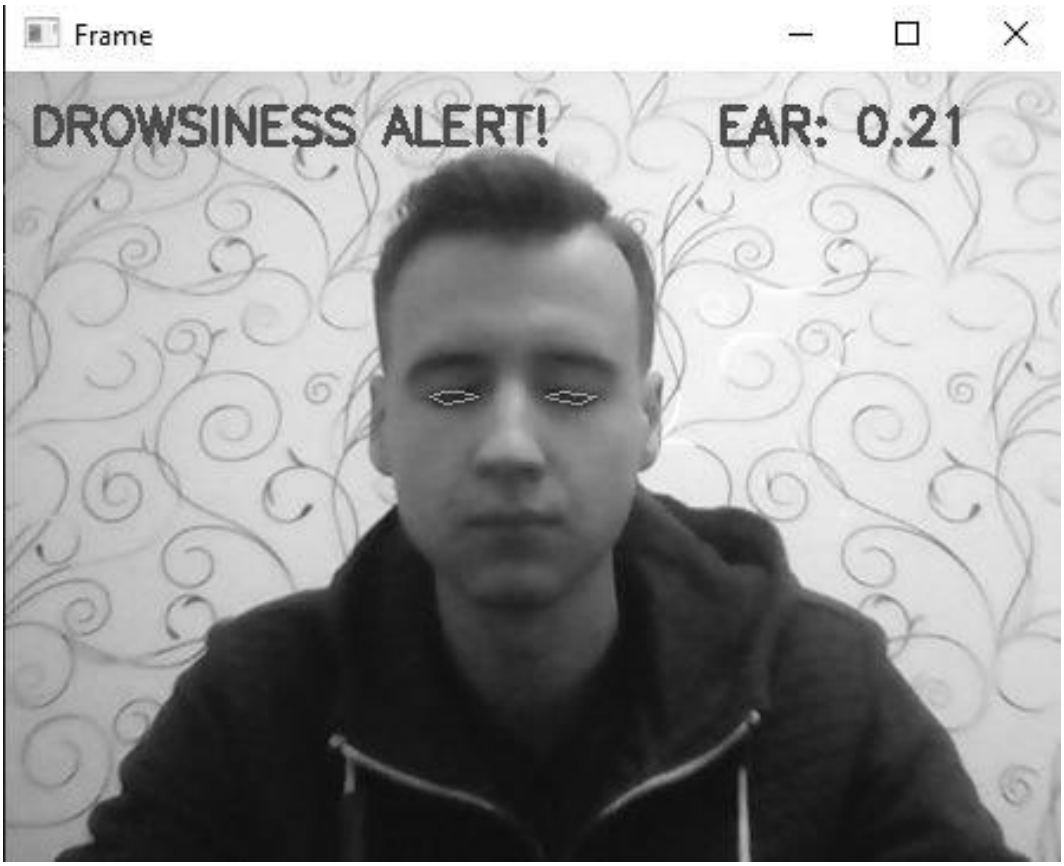

Рис. 3.14 - Виклик звукового повідомлення для пробудження водія

Після того, як водій знову відкрив очі, ознаки обличчя перевіряються повторно. Програмні константи і змінні повертають значення за замовчуванням.

Якщо кількість кадрів, у яких людина заплющила очі, перевищує FRAMES_EAR, необхідно видати звуковий сигнал.

Експериментально було виявлено, що значення THRESHOLD_EAR, що дорівнює 0,3, добре працює у різних ситуаціях.

Значення FRAMES_EAR також експериментально було встановлено на 48 – якщо людина заплющує очі на 48 послідовних кадрів, то відтворюватиметься звуковий сигнал. Можна зробити детектор більш чутливим, зменшивши FRAMES_EAR – аналогічно, менш чутливим, збільшивши його. Розрахунок евклідових відстаней відбувається миттєво і не залежить від хмарних баз даних. Це дає перевагу пропонованої системи над вже існуючими системами. Швидкодія роботи системи відповідає поставленим задачам.

Система має ризик того, що перевтомлений водій не буде зволікати на звукове повідомлення системи. Пропонується додатково повідомляти водія про заплющенні очі вібрацією водійського крісла та зменшенням обертів двигуна.

3.4 Висновки до розділу

У розділі приведена практична реалізація удосконалення методів розпізнавання стану очей водія під час дорожнього руху. Описані основне обладнання для системи і інструменти для повноцінної розробки системи. Наведено алгоритм роботи програми. Приведено графік порогів стану очей водія, отриманого завдяки обчисленням евклідової відстані між наборами ознак очей, отриманих завдяки індексованого списку бібліотеки dlib. Отже, детектування стану очей водія відбувається на постійній основі, а відсутність постійного зв’язку з базами даних удосконалює швидкодію алгоритму.

Оцінка пропонованої системи задовольняє результатам поточного етапу розробки завдань, сформованих на початку цього етапу. Система відповідає поставленим вимогам. Оцінені певні ризики та приведені варіанти їх вирішення. Завдяки покроковому тестуванню системи перевірені на працездатність всі окремі функції системи. Помилок і некоректної роботи в системі не знайдено. Тести на швидкодію системи, завдяки визначенню часу від початку роботи системи до кінцевого етапу, показали очікуванні результати. Пропонована система удосконалює роботу існуючих систем.

ВИСНОВКИ

В ході кваліфікаційної роботи магістра, проаналізована інформація існуючих передових методів машинного зору та розпізнавання облич, і систем безпеки водія в яких вони використовуються. Були проаналізовані актуальні проблеми, які потребують вирішення і удосконалення методів детектування обличчя водія та розпізнавання стану його очей. Запропоновано удосконалення методу, який базується на алгоритмі детектування очей водія без використання бази даних, зв’язків з мережею та з використанням інфрачервоної камери для запобігання перешкод відстеження очей водія під час дорожнього руху. На основі бібліотеки машинного зору, а саме розпізнавання обличчя, побудовано алгоритм роботи пропонованої системи. На основі алгоритму змодельовано удосконалений метод розпізнавання очей водія для уникнення аварійних ситуацій. В результаті вирішення поставлених задач дослідження, покращено швидкодію роботи методу без втрати точності його роботи.

СПИСОК ЛІТЕРАТУРИ

[1]. Система контролю втоми водія [Електронний ресурс] – Режим доступу до ресурсу: http://systemsauto.ru/active/drowsiness_detection_system.html

[2]. Засипання за рулем [Електронний ресурс] – Режим доступу до ресурсу: https://salavatmed.ru/content/zasypanie-za-rulem-yavlyaetsya-prichinoy-okolo-20-dorozhno-transportnyh-proisshestviy-dtp

[3]. Технологія Seeing machines [Електронний ресурс] – Режим доступу до ресурсу: https://www.seeingmachines.com/technology/

[4]. Моніторинг уваги водія DS [Електронний ресурс] – Режим доступу до ресурсу: https://www.dsautomobiles.co.uk/inside-ds/ds-news/ds-automobiles-anti-fatigue-technology

[5]. Cadillac CT6 Super Cruise 2018 [Електронний ресурс] – Режим доступу до ресурсу: https://www.motortrend.com/reviews/2018-cadillac-ct6-super-cruise-review/

[6]. Система Driver Alert Control, DAC від Volvo [Електронний ресурс] – Режим доступу до ресурсу: http://systemsauto.ru/active/drowsiness_detection_system.html

[7]. Системи розпізнавання облич Facial recognition technology (FRT) [Електронний ресурс] – Режим доступу до ресурсу: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BB%D0%B8%D1%86_(Facial_recognition)

[8]. Аналіз існуючих підходів до розпізнавання облич [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/synesis/blog/238129/

[9]. Виявлення сонливості водія [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/581318/

[10]. DLib: бібліотека для машинного навчання [Електронний ресурс] – Режим доступу до ресурсу: https://www.kdnuggets.com/2014/06/dlib-library-machine-learning.html

[11]. Розпізнавання облич - версія Dlib (чотири) [Електронний ресурс] – Режим доступу до ресурсу: https://russianblogs.com/article/1081850334/

[12]. Евклідова відстань [Електронний ресурс] – Режим доступу до ресурсу: http://statistica.ru/glossary/general/evklidovo-rasstoyanie/

[13]. Евклідова група - Euclidean group [Електронний ресурс] – Режим доступу до ресурсу: https://livepcwiki.ru/wiki/Euclidean_group

[14]. Python [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Python

[15]. Playsound 1.3.0 [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/playsound/

[16]. Dlib C++ library [Електронний ресурс] – Режим доступу до ресурсу: http://dlib.net/

[17]. numpy 1.21.4 [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/numpy/

[18]. PyCharm IDE для професійної розробки на Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/pycharm/

[19]. OpenCV Комп'ютерний зір із відкритим вихідним кодом [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/4.x/d1/dfb/intro.html

[20]. Тестування програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F

[21]. Тестування програмного забезпечення - основні поняття та визначення [Електронний ресурс] – Режим доступу до ресурсу: http://www.protesting.ru/testing/

Додаток А

Лістинг програми для детектування стану очей водія

Код програми:

```
import dlib

import numpy

import cv2

from playsound import playsound

import time



def runAlarm(path):
        # відтворити повідомлення
playsound.playsound(path)



def findEAR(eye):
         # обчислити евклідові відстані між двома наборами
         # вертикальних ознак ока (x, y)-координат
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
        # те саме для горизонтальних ознак
```

```
    C = dist.euclidean(eye[0], eye[3])
        # обчислити EAR
    ear = (A + B) / (2.0 * C)
        # повернути його
    return ear

        # визначити дві константи:
        # одну для співвідношення сторін ока, щоб визначити моргання,
        #а потім другу константу для кількості послідовних кадрів.
        # око має бути нижче порогового значення, щоб спрацював сигнал тривоги

THRESHOLD_EAR = 0.3
FRAMES_EAR = 48

        # ініціалізувати лічильник кадрів, а також логічне значення,
        # використовуване для індикації спрацьовування сигналізації

COUNTER = 0
ALARM_ON = False

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

(leftEyeSt, leftEyeEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rightEyeSt, rightEyeEnd) = face_utils. FACIAL_LANDMARKS_IDXS["right_eye"]

        # запустити захоплення відеопотоку з веб-камери
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)
```

```
цикл з кадрів з відеопотоку
while True:
	# захопити кадр із потокового відеофайлу, змінити його розмір
	# і перетворити на канали у відтінки сірого
frame = vs.read()
frame = imutils.resize(frame, width=450)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

	# Виявити обличчя за допомогою детектора
rects = detector(gray, 0)
```