НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
імені ІГОРЯ СІКОРСЬКОГО

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено
Завідувач кафедри
_____ Романкевич В. О.

"___"_____2023р.

Дипломний проєкт

на здобуття ступеня бакалавра
зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Програмна система виявлення сонливості водія на основі OpenCV»

Виконав:
студент 4 курсу, групи КВ-92
Гула Михайло Романович _____

Керівник доц.каф.СПСКС, к.т.н. Петрашенко А. В, _____

Консультант з нормоконтролю,
доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____

Рецензент:_____ _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Романкевич В.О.

"___"_____2023р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА

Гули Михайла Романовича

1. Тема проєкту «Програмна система виявлення сонливості водія на основі OpenCV»,
   керівник проєкту доц.каф.СПСКС, к.т.н. Петрашенко Андрій Васильович, затверджені наказом по університету від «31» травня 2023 р. №2107-С

2. Строк подання студентом роботи: "19" червня 2023 р.

3. Вхідні дані для дипломної роботи Технічне завдання

4. Зміст пояснювальної записки
   1) Аналіз існуючих систем контролю втоми водія та обґрунтування теми дипломного проекту
   2) Огляд існуючих алгоритмів програмного забезпечення для виявлення рівня втомленості водія
   3) Проектування та розробка програмного застосунку для виявлення втомленості водія
   4) Тестування розробленої програмної системи розпізнавання втомленості водія

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
   1) Діаграма варіантів використання програмної системи
   2) Діаграма послідовності програмної системи
   3) Структурна схема алгоритму виявлення сонливості водія
   4) Структурна схема алгоритму головних компонент

6. Консультанти:

Питання	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконроль	Клятченко Я. М., доцент		

7. Дата видачі завдання: 07.11.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строквиконання етапів	Примітка
1.	Вивчення літератури за тематикою роботи та збір даних	07.11.2022	
2.	Підготовка матеріалів першого розділу дипломної роботи	20.11.2022	
3.	Створення архітектури програмного комплексу	05.05.2023	
4.	Підготовка матеріалів другого розділу дипломної роботи	10.05.2023	
5.	Підготовка матеріалів третього розділу дипломної роботи	18.05.2023	
6.	Підготовка матеріалів четвертого розділу дипломної роботи	22.05.2023	
7.	Підготовка матеріалів п’ятого розділу дипломної роботи	24.05.2023	
8.	Підготовка графічної частини дипломної роботи	25.05.2023	
9.	Підготовка матеріалів графічної частини проєкту	16.05.2023	
10.	Оформлення технічної документації проєкту	01.06.2023	

Студент _____ **Гула М.Р.**

Керівник роботи _____ **Петрашенко А.В.**

АНОТАЦІЯ

Бакалаврський дипломний проєкт включає пояснювальну записку (61 стор., 28 рис., список використаної літератури з 16 найменувань, 3 додатків) та презентацію (15 слайдів).

Об’єкт розробки – створення програмної системи виявлення рівня втомленості водія транспортного засобу за кермом з використанням сучасних технологій комп’ютерного зору та машинного навчання.

Програмна система дозволяє виявляти та розпізнати патерни руху очей водія; виявляти рівень зачиненості очей; сповіщати водія про наявну небезпечну ситуацію. В процесі розробки були використані технології машинного навчання, а саме OpenCV, Tensorflow, Keras, NumPy, TorchVision.

В ході розробки програмної системи:

– Проведено аналіз предметної області виявлення сонливості водіїв.

– Систематизовано основні аналоги системи розпізнавання сонливості водія, виділено основні переваги та недоліки існуючих систем.

– Розглянуто основні архітектурні патерни розробки системи розпізнавання втомленості водія та виділено основі переваги і недоліки.

– Спроєктовано нейронну мережу для розпізнавання втомленості водія.

– Розроблено веб-сервіс для подальшої інтеграції розробленої нейронної мережі для спрощення взаємодії кінцевого користувача з системою.

Результатом роботи є програмна система, яка здатна виявляти ознаки сонливості водія на основі обробки зображень з використанням OpenCV. Система може виявляти замкнуті очі, неправильну позицію голови та інші ознаки сонливості. При виявленні таких ознак система може надсилати попередження водієві, що дозволить забезпечити безпеку на дорозі.

Ключові слова: програмна система, виявлення сонливості, OpenCV, комп’ютерний зір, машинне навчання, TorchVision, дорожній рух.

ABSTRACT

The bachelor's diploma project includes an explanatory note (61 pages, 28 figures, a list of 16 references, 3 appendices) and a presentation (15 slides).

The object of the development is the creation of a software system for detecting the level of fatigue of a vehicle driver while driving using modern technologies of computer vision and machine learning.

The software system allows you to detect and recognize the driver's eye movement patterns; detect the level of eye closure; notify the driver about the existing dangerous situation. In the development process, machine learning technologies were used, namely OpenCV, Tensorflow, Keras, NumPy, TorchVision.

During the development of the software system:

– An analysis of the subject area of driver drowsiness detection was carried out.

– Systematized the main analogues of the driver drowsiness recognition system, highlighted the main advantages and disadvantages of the existing systems.

– The main architectural patterns of the development of the driver recognition system were considered and the advantages and disadvantages were highlighted.

– A neural network was designed to recognize driver fatigue.

– A web service was developed for further integration of the developed neural network to simplify the interaction of the end user with the system.

The result of the work is a software system capable of detecting signs of driver drowsiness based on image processing using OpenCV. The system can detect closed eyes, incorrect head position and other signs of drowsiness. If signs are detected, the system can send a warning to the driver, which will ensure safety on the road.

Keywords: software system, drowsiness detection, OpenCV, computer vision, machine learning, TorchVision, road traffic.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.002 ТЗ	Програмна система виявлення	4		
			сонливості водія			
			на основі OpenCV.			
			Технічне завдання.			
	А4	ІАЛЦ.045490.003 ТП	Програмна система виявлення	2		
			сонливості водія			
			на основі OpenCV.			
			Відомість технічного проєкту.			
	А4	ІАЛЦ.045490.004 ПЗ	Програмна система виявлення	61		
			сонливості водія			
			на основі OpenCV.			
			Пояснювальна записка.			
	А4	ІАЛЦ.045490.005 Д1	Схема роботи алгоритму	1		
			виявлення сонливості водія			
			Схема алгоритму системи.			

Змін.	Арк.	№ докум.	Підпис	Дата	**ІАЛЦ.045490.001 ОА**		
Розробив		Гула М. Р.			Програмна система виявлення сонливості водія на основі OpenCV ***Опис альбому***	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А. В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-92	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.006 Д2	Діаграма послідовності	1		
			програмної системи.			
			Діаграма послідовності.			
	А4	ІАЛЦ.045490.007 Д3	Діаграма варіантів	1		
			використання системи.			
			Діаграма прецедентів.			
	А4	ІАЛЦ.045490.008 Д4	Структурна схема алгоритму	1		
			головних компонент.			
			Схема алгоритму системи.			
		Диск CD-ROM	Текст пояснювальної записки.			
			Архівований код програми			
			у форматі .zip.			
			Презентація дипломного проєкту.			
			Графічний матеріал роботи.			

ТЕХНІЧНЕ ЗАВДАННЯ

До дипломного проєкту

на тему: «Програмна система виявлення сонливості водія на основі OpenCV»

Київ – 2023

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмна система виявлення сонливості водія на основі OpenCV».

Галузь застосування: Виявлення ознак сонливості водія, що може свідчити про зменшення його концентрації під час керування транспортним засобом.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка програмної системи виявлення сонливості водія з використанням OpenCV та технологій машинного навчання.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

1. сумісність з операційними системами (MacOS, Linux, Windows);
2. виявлення обличчя водія транспортного засобу.
3. виявлення та відстеження руху очей водія транспортного засобу.
4. вимірювання позиції очей водія транспортного засобу.
5. надійність та швидкодійність при виявленні сонливості.

6. адаптивність до різних умов (наприклад, освітлення/фізичні характеристики водія транспортного засобу).

7. інтеграція з іншими системами (системи контролю за увагою водія, системи допомоги при тривозі або системи автопілотування).

8. наявність системи упередження та реагування.

5.2 Вимоги до апаратного забезпечення

1. Процесор: 2,8-ядерний, Intel Core I3/I5/I7, AMD Ryzen 3/5/7;
2. Оперативна пам’ять: 8 Гб;
3. Наявність доступу до мережі Internet (LAN, WI-FI);

5.3 Вимоги до програмного та апаратного забезпечення користувача

1. Операційна система Windows 10/11, MacOS 12.0+ або Linux.
2. Наявність камери з високоякісним зображенням та освітленням.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1	Вивчення літератури за тематикою проекту	15.04.2023
2	Розроблення та узгодження технічного завдання	30.04.2023
3	Аналіз існуючих рішень	05.05 2023
4	Підготовка матеріалів першого розділу дипломного проекту	10.05.2023
5	Підготовка матеріалів другого розділу дипломного проекту	18.05.2023
6	Підготовка графічної частини дипломного проекту	20.05.2023
7	Оформлення документації дипломного проекту	25.05.2023
8	Попередній огляд матеріалів диплому на кафедрі	30.05.2023

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.004 ПЗ	Програмна система виявлення	61		
			сонливості водія			
			на основі OpenCV.			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Схема роботи алгоритму	1		
			виявлення сонливості водія			
			Схема алгоритму системи.			
	А4	ІАЛЦ.045490.006 Д2	Діаграма послідовності	1		
			програмної системи.			
			Діаграма послідовності.			
	А4	ІАЛЦ.045490.007 Д3	Діаграма варіантів	1		
			використання системи.			
			Діаграма прецедентів.			
	А4	ІАЛЦ.045490.008 Д4	Структурна схема алгоритму	1		
			головних компонент.			
			Схема алгоритму системи.			

Змін.	Арк.	№ докум.	Підпис	Дата	**ІАЛЦ.045490.003 ТП**		
Розробив		Гула М. Р.			Програмна система виявлення сонливості водія на основі OpenCV ***Відомість технічного проєкту***	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко А. В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-92	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Текст пояснювальної записки.			
			Архівований код програми			
			у форматі .zip.			
			Презентація дипломного проєкту.			
			Графічний матеріал роботи.			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП	Арк. 2

ПОЯСНЮВАЛЬНА ЗАПИСКА

До дипломного проєкту

на тему: «Програмна система виявлення сонливості водія на основі OpenCV»

Київ – 2023

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

- ІАЛЦ.045490.005 Д1. Діаграма варіантів використання програмної системи
- ІАЛЦ.045490.006 Д2. Діаграма послідовності програмного забезпечення
- ІАЛЦ.045490.007 Д3. Схема роботи алгоритму виявлення втомленості водія
- ІАЛЦ.045490.008 Д4. Структурна схема алгоритму головних компонент

Додаток 2. ІАЛЦ.045490.009 Д5. Презентація

Додаток 3. ІАЛЦ.045490.010 Д6. Лістинг програми

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

EAR (Eye Aspect Ratio) – скалярне значення, яке реагує, зокрема на відкриття та закриття очей.

CNN (Convolutional neural network) – клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень. CNN взяли за основу з'єднання нейронів зорової кори тварин.

ADAS (Advanced driver-assistance systems) – електронна система, що допомагає водію керувати автомобілем і парковкою. Завдяки безпечному людино-машинному інтерфейсу сприяє безпеці автомобілей і дорожнього руху.

LBP (Local binary patterns) – тип візуального дескриптора, який використовується для класифікації в комп'ютерному зорі.

SUSAN (Smallest Univalue Segment Assimilating Nucleus) – є основою для алгоритмів для визначення країв, кутів і зменшення структури шуму зображення.

ВСТУП

Виявлення сонливості водія – це технологічний прогрес у сфері безпеки транспортних засобів, який спрямований на зменшення аварій, спричинених керування у стані сну. Згідно з дослідженнями, втома є причиною майже 20% усіх дорожньо-транспортних пригод і до 50% аварій, які трапляються на окремих дорогах. Для виявлення сонливості водія доступні різні технології. Одним із способів є моніторинг моделі керування кермом, яка в основному передбачає використання керма від електричної системи керування. Однак цей метод працює лише тоді, коли водій активно керує транспортним засобом, а не покладається на автоматичну систему утримання в смузі руху. Інший підхід полягає у перевірці положення автомобіля під час контролю смуги руху, що вимагає використання камери моніторингу смуги руху. Знову ж таки, цей метод можна використовувати лише тоді, коли водій активно керує транспортним засобом. Нарешті, методи комп'ютерного зору можна використовувати для спостереження за очима та обличчям водія за допомогою вбудованої камери або мобільних пристроїв. Ця технологія дозволяє постійно стежити за станом водія та може виявляти ознаки сонливості чи втоми.

Удосконалення технологій, спрямованих на виявлення та запобігання сонливості під час керування автомобілем, створює серйозну проблему для вдосконалення систем запобігання аварій. Вкрай важливо попередити водія в момент виявлення сонливості, щоб запобігти будь-яким потенційним нещасним випадкам. Стан очей водія може бути засобом виявлення сонливості, тому необхідно уважно стежити за будь-якими ознаками втоми. Ця технологія може відігравати життєво важливу роль у забезпеченні безпеки водія на дорозі, особливо під час тривалих поїздок. Реалізація системи виявлення сонливості водія може використовувати бібліотеки мови програмування Python. Ці

бібліотеки дозволяють системі аналізувати та визначати тривалість закритих очей водія, що може свідчити про сонливість. Коли очі закриті протягом тривалого періоду часу, система подає сигнал тривоги, щоб попередити водія та привернути увагу, запобігаючи можливим нещасним випадкам, спричиненим водінням у стані втоми.

Отже, актуальність теми кваліфікаційної роботи зумовлена недосконалістю сучасних технологій детектування стану оцей водія для уникнення аварійних ситуацій з використанням машинного навчання.

Метою кваліфікаційної роботи є розробка і вдосконалення методу виявлення рівня втомленості (сонливості) водія за кермом з використанням сучасних технологій комп'ютерного зору та машинного навчання.

Для досягнення мети кваліфікаційної роботи необхідно виконати наступні завдання:

- Проаналізувати існуючі сучасні методи комп'ютерного зору і розпізнавання облич та систем безпеки водія, в яких вони використовуються.
- Спроектувати метод виявлення рівня втомленості водія, покращивши швидкодію роботи існуючих алгоритмів без втрати точності їх роботи.
- Розробити застосунок для виявлення рівня втомленості водія для використання в системах безпеки з використанням веб-камери.

Об'єктом дослідження є процес виявлення рівня втомленості водія на основі даних з внутрішньої веб-камери.

Предметом дослідження є методи виявлення рівня втомленості з внутрішньої веб-камери на основі машинного навчання.

Методи дослідження включають методи комп'ютерного зору, теорії алгоритмів, методи машинного навчання, методи програмування.

1. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ КОНТРОЛЮ ВТОМИ ВОДІЯ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Проблема і потенційні наслідки втомленості водіїв

Втома водія є суттєвою причиною приблизно 20% усіх серйозних дорожньо-транспортних пригод, головним наслідком яких є засинання за кермом. Довгі поїздки, особливо вночі та монотонними дорогами, становлять найбільший ризик того, що водії заснуть.

Дослідження [12, 14] демонструють, що час реакції водія значно скорочується вже після чотирьох годин безперервної їзди, а через вісім годин – у шість разів. Варто зазначити, що на відміну від поїздів і літаків, де використовується автоматизація, водіння вимагає постійної пильності. Навіть кілька секунд сну можуть призвести до смертельної аварії. Відомо, що дорожньо-транспортні пригоди, які відбуваються внаслідок того, що водії засинають за кермом, мають летальні наслідки та призводять до серйозних травм. Однією з основних причин серйозності таких інцидентів є відсутність здатності водія вжити відповідних заходів, наприклад, знизити швидкість транспортного засобу або вжити заходів для ухилення від перешкод.

Згідно з дослідженням [11], основною причиною того, що водії несподівано засинають під час водіння, є накопичення недосипу або перевтоми, що призводить до стану «безсоння» під час поверхневого сну. Найпоширенішою причиною денної сонливості, що призводить до таких явищ, є синдром обструктивного апное сну (СОАС).

Статистика показує, що рівень поширеності СОАС серед осіб віком 30 років і старше становить 4-7%. Більш того, приблизно у 2% дорослого населення розвивається важка форма синдрому. Це означає, що щодня сотні тисяч водіїв,

які страждають від сильної денної сонливості, керують транспортними засобами на дорогах, що збільшує ризик аварій.

Система контролю втоми відповідає за моніторинг фізичного стану водія з метою забезпечення його безпеки. Вона виявляє будь-які відхилення від очікуваного стандарту та попереджає водія про необхідність зупинитися та зробити перерву, якщо це необхідно. Залежно від методу оцінки втоми водія доступні три різні типи систем. Перший спосіб контролює дії водія, а другий – рух автомобіля. Нарешті, третій метод покладається на спостереження за поглядом водія, щоб виявити будь-які ознаки сонливості чи втоми. На рисунку 1.1 представлено візуалізацію спостереження за станом водія.

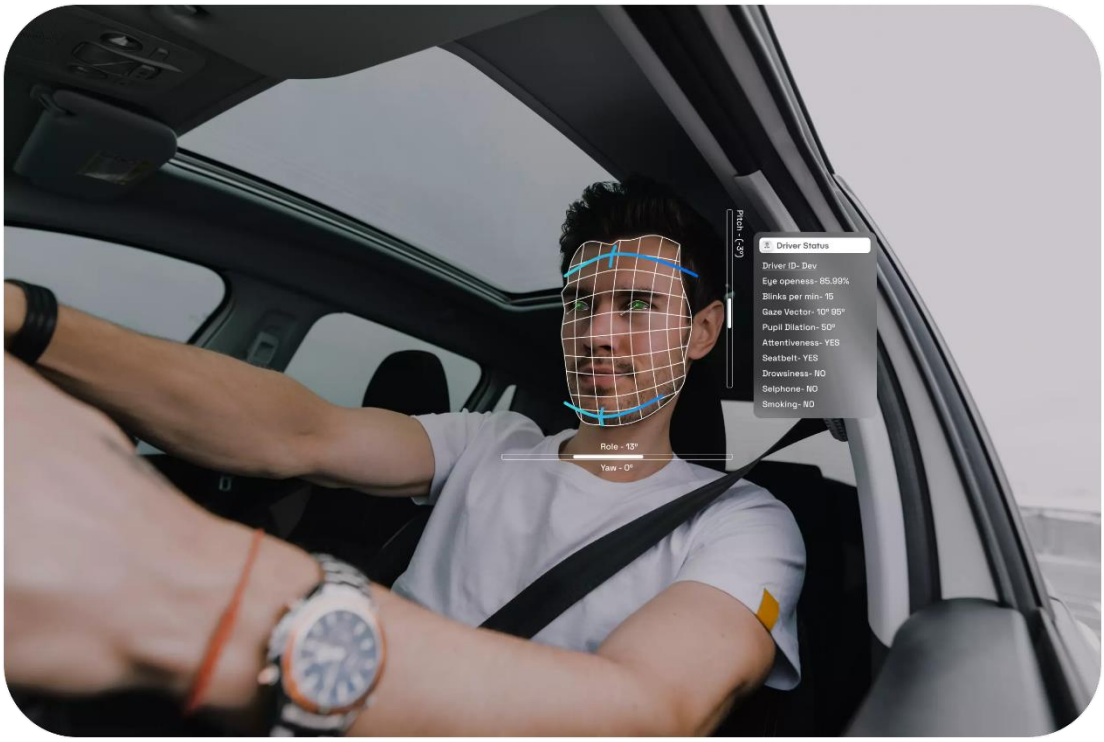


Рисунок 1.1 – Спостереження за станом втомленості водія

Сонливість – це стан, який класифікується як психічний розлад, викликаний нестачею сну. Це не те, з чим можна впоратися просто «прокинувшись»; єдиний спосіб подолати це – відпочити. Водіння в стані сонливості вже давно викликає занепокоєння практично на всіх ринках

транспортних засобів, оскільки водіння в стані сонливості може призвести до нещасних випадків через погіршення когнітивних і фізичних здібностей. На відміну від алкоголю, який можна виявити за допомогою алкотестера, наразі не існує надійних методів точного вимірювання рівня сонливості водія.

Компанія Seeing Machines провела масштабне дослідження, яке охоплює більше десяти років, щоб розробити ефективний метод оцінки ризику втрати водієм контролю над автомобілем через сонливість [9]. Поєднавши найсучасніші базові алгоритми відстеження погляду з передовими методами машинного навчання та аналізуючи ретельно зібрані та підібрані набори даних, вони створили сигнал рівня сонливості водія (DDL). Цей сигнал забезпечує надійну та точну оцінку рівня сонливості водія та ризику, пов’язаного з керуванням у найближчому майбутньому [10].

Розробка умовно автономних транспортних засобів і розширених функцій безпеки, таких як системи запобігання зіткненням, вимагає точного визначення рівня участі водія в управлінні завданням або сценарії руху, як показано на рисунку 1.2. Це вимірювання допомагає системам транспортного засобу передавати контроль або відновлювати повноваження від водія безпечним і інтелектуальним способом, як описано в [12].

Рисунок 1.2 – Автономне керування транспортним засобом

1.2 Огляд існуючих систем контролю втоми водія

Запровадження нових законів, прийнятих на початку цього року, з 2023 року зробить попередження про сонливість водія та відволікання уваги обов'язковим для всіх нових автомобілів у всьому ЄС. Наприклад, з 2015 року у Великій Британії сталося 4000 аварій і 150 загиблих через втому водія [2].

DS Automobiles випереджає вимоги щодо розвитку безпеки завдяки системі "DS DRIVER ATTENTION MONITORING", яка допомагає тримати водіїв активними та не дозволяє їм заснути за кермом. Передова технологія відстежує ознаки втоми, потенційно допомагаючи врятувати близько 50 життів на рік на дорогах Великобританії. Система " DS DRIVER ATTENTION MONITORING" складається з відеокамери, встановленої над кермом і верхньою частиною лобового скла, яка постійно стежить за очима водія. Вона відповідає за вивчення та аналіз послідовності миготіння, рухів, зроблених людиною через голову або напрямок погляду. Особливість системи для контролю полягає в тому, що її можна використовувати як вдень, так і вночі, крім того, інфрачервоні датчики можуть проходити через сонцезахисні окуляри, щоб виконувати свою функцію, серед яких виділяється попередження, коли людина відволікає увагу від дороги на ваш телефон, оскільки система розцінює цю дію як несправність і негайно відображає попередження на центральному екрані автомобіля, а потім звуковий сигнал.

Функція моніторингу положення автомобіля постійно контролює положення автомобіля відносно дорожньої розмітки та попереджає водія звуковим сигналом про будь-які несподівані або раптові рухи керма. Це забезпечує безпеку водія, надаючи попередження, якщо транспортний засіб з будь-якої причини відхиляється від наміченого шляху [4].

Зовнішній вигляд концепції функціональних можливостей вищеописаної системи моніторингу представлений на рисунку 1.3.

Рисунок 1.3 – Концепція моніторингу уваги та концентрації водія [6]

Cadillac Super Cruise, як продемонстровано на рисунку 1.4, є новою технологією, яка включає адаптивний круїз-контроль. Порівняно з Autopilot від Tesla та ProPilot Assist від Nissan, унікальний аспект Cadillac Super Cruise полягає у використанні карт високої роздільної здатності для визначення місцезнаходження транспортного засобу з високою точністю, у межах лише чотирьох дюймів. Завдяки відображенню місцевості на відстані до 1,5 милі попереду Super Cruise забезпечує кращу обізнаність водія про ситуацію, підвищуючи безпеку на дорозі. HD-карти, які використовує Cadillac Super Cruise, періодично оновлюються щокварталу через систему GM OnStar.

Рисунок 1.4 – Візуалізація роботи системи Cadillac Super Cruise [8]

На відміну від інших систем ADAS, таких як Tesla Autopilot і ProPilot Assist, які вимагають, щоб руки водія залишалися на кермі, щоб система продовжувала працювати, Super Cruise залишається активним, лише якщо водій уважний, що забезпечує максимальну безпеку. Інфрачервона камера є необхідним компонентом системи, що забезпечує механізм виявлення відволікання або неуважності водія при використанні системи управління.

Далі, розглянемо систему Mercedes Attention Assist. Дана система здатна підвищити безпеку руху, особливо у далеких та нічних поїздках. Система Attention Assist за характером водіння розпізнає типові ознаки втоми та наростаючої неуважності водія, попереджаючи його за допомогою візуальних та звукових сигналів про загрозу секундного засинання. Система Attention Assist при швидкості руху в діапазоні від 80 до 180 км/год постійно стежить за процесом водіння та поворотами кермового колеса. Помічаючи появу певних прийомів водіння, які часто виникають при перевтомі, вона подає візуальне

попередження через комбінацію приладів у вигляді символу, що зображує чашку кави, а також попереджувальні акустичні сигнали. На рисунку 1.5 представлена візуалізація приладової панелі системи Mercedes Attention Assist.

Рисунок 1.5 – Візуалізація роботи системи Mercedes Attention Assist [10]

Система Driver Alert Control, зображена на рисунку 1.6, є функцією безпеки, розробленою компанією Volvo, яка відрізняється від системи Attention Assist щодо свого функціонування. Driver Alert Control контролює рух автомобіля на дорозі, використовуючи відеокамеру, спрямовану вперед, щоб відстежувати положення автомобіля в межах смуги руху. Система призначена для виявлення будь-яких відхилень від заданих параметрів руху, що інтерпретується як настання втоми водія. Система DAC здатна видавати два рівні попередження, а саме «м'який» і «жорсткий», залежно від тяжкості стану водія. Звуковий сигнал, який створює система, змінюється за гучністю та тоном між двома рівнями попередження. Варто зазначити, що система попередження про виїзд зі смуги руху відіграє вирішальну роль у функціонуванні системи DAC, оскільки остання базується на принципах конструкції першої. Однак важливо зазначити, що система DAC працює лише тоді, коли автомобіль рухається зі швидкістю 60 км/год або вище.

Рисунок 1.6 – Візуалізація роботи системи Volvo Driver Alert Control [10]

General Motors впровадила систему під назвою Seeing Machines для оцінки втоми водія, яка використовує технологію контролю очей. Система Seeing Machines включає спеціальний датчик, який контролює ступінь відкриття очей водія та напрямок погляду, як показано на рисунку 1.7.

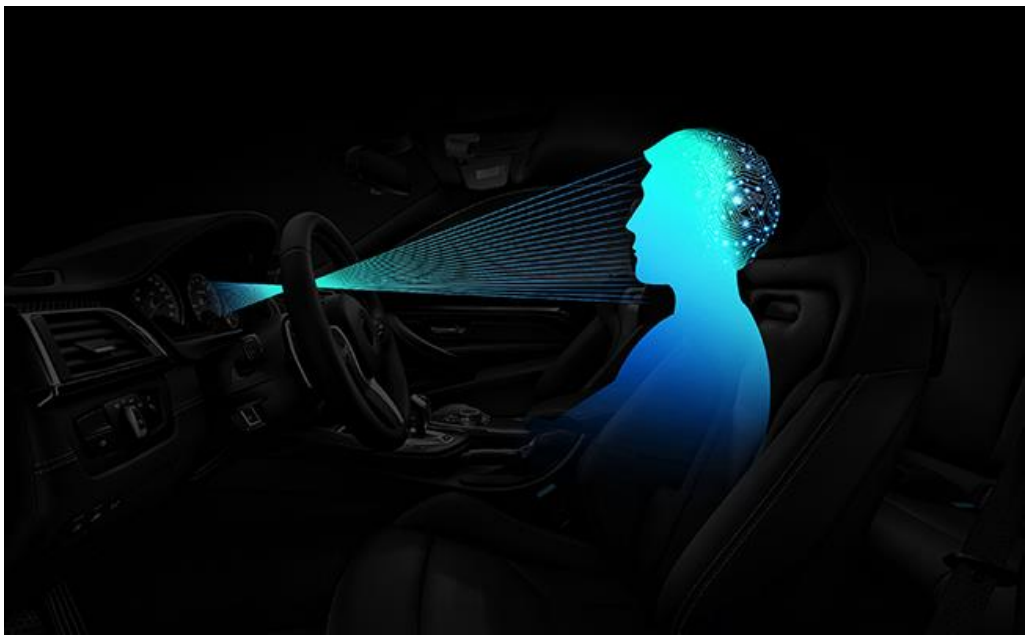


Рисунок 1.7 – Візуалізація роботи системи DAC Seeing Machines [10]

1.3 Огляд існуючих методів виявлення втомленості водія

Розпізнавання обличчя – це тип програмного забезпечення, яке автоматично ідентифікує людські обличчя на зображеннях або відео, а також може визначити особу людини, порівнюючи її унікальні контури обличчя з інформацією в базі даних. Ця технологія викликала значний інтерес через велику кількість завдань, які вона може виконувати.

Зокрема, розпізнавання обличчя передбачає використання біометричного програмного забезпечення, яке аналізує чіткі контури обличчя людини, показані на рисунку 1.8, для перевірки або ідентифікації її особистості. Оскільки структура обличчя кожної людини унікальна, спеціалізоване програмне забезпечення може проаналізувати ці шаблони та зіставити їх із інформацією бази даних для подальшої ідентифікації особи.

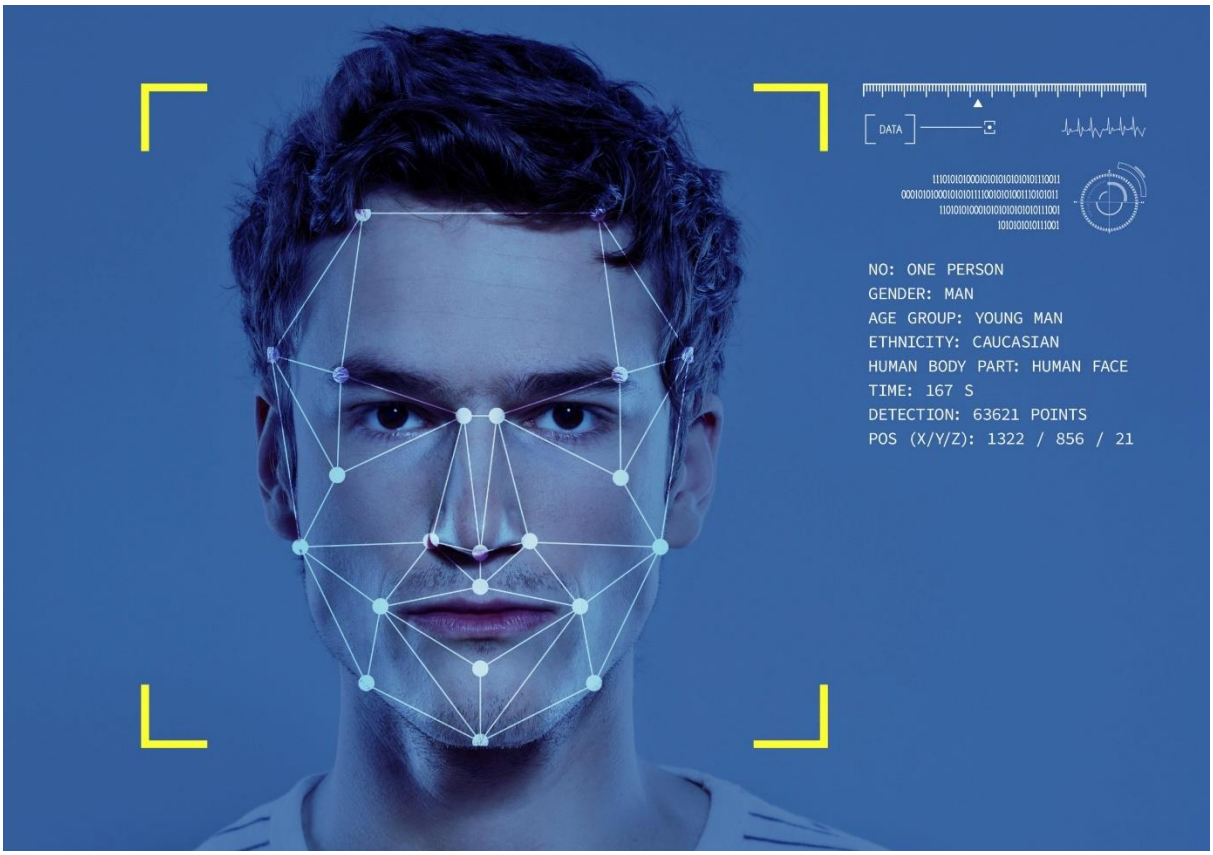


Рисунок 1.8 – Приклад біометричної ідентифікації людини з використанням технологій розпізнавання обличчя [12]

Технологія розпізнавання облич має декілька недоліків, оскільки вона може бути не настільки ефективною за певних умов. Наприклад, якість розпізнавання погіршується в умовах поганого освітлення або коли змінюється положення голови чи ракурс. Існують різні підходи, що використовуються для створення алгоритмів розпізнавання облич, одним із яких є емпіричний підхід. Цей метод передбачає застосування схожих на людину правил для ідентифікації людини, таких як визначення таких ознак, як яскравість лоба та рівномірність кольору та яскравості в центральній частині обличчя, а також наявність частин обличчя, таких як ніс, рот і очі на зображенні. Для ідентифікації осіб на зображеннях значно зменшується область, де очікується поява людини, або створюються перпендикулярні гістограми. Хоча ці методи досить прості у застосуванні, вони не завжди ефективні, особливо коли на задньому плані багато об’єктів, у кадрі кілька людей або змінюється ракурс зображення. На рисунку 1.9 представлений приклад емпіричного методу розпізнавання обличчя для ідентифікації.

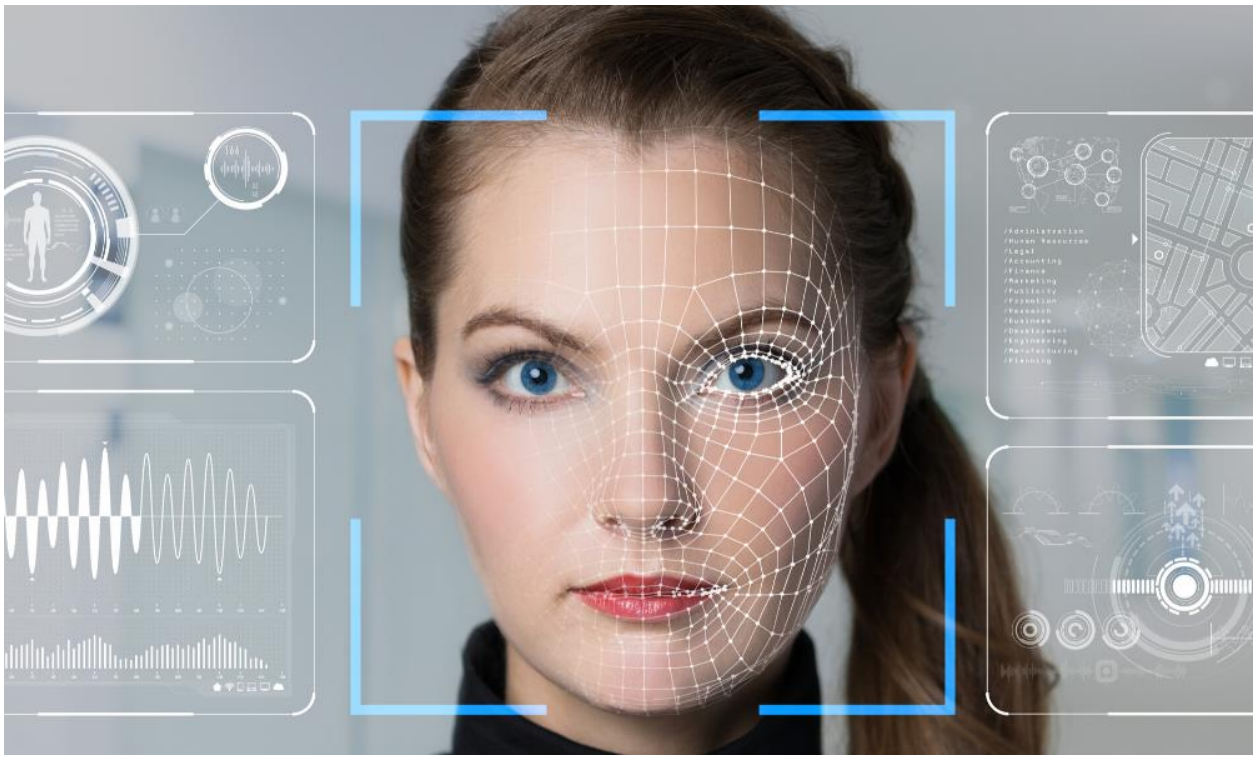


Рисунок 1.9 – Візуалізація емпіричного методу розпізнавання обличчя

Наступний метод розпізнавання обличчя передбачає використання інваріантних ознак, які є унікальними та стабільними рисами обличчя, які використовуються для ідентифікації людей. Цей метод заснований на емпіризмі та намагається імітувати те, як люди розпізнають обличчя. Він визначає характерні ділянки обличчя, такі як його контури, варіації форми та контрасту, і поєднує всі ці ознаки для перевірки особи людини. Цей підхід ефективний навіть тоді, коли людина повернута головою, але він може не спрацювати, якщо на задньому плані є інші люди або інша обстановка.

Наступний метод, який використовується для виявлення особи, включає шаблони, які попередньо визначені розробником. Ці шаблони представляють стандарт або шаблон для зовнішнього вигляду людини, і алгоритм прагне ідентифікувати та зіставити цей шаблон з кожним сегментом зображення, незалежно від його кута чи масштабу. Хоча така система вимагає великої обчислювальної потужності, сучасні технології розпізнавання облич еволюціонували, щоб використовувати тестові зображення як засіб навчання. Це передбачає використання баз даних зображень, що містять як зображення людей, так і нелюдей, для навчання системи.

На рисунку 1.10 представлений приклад бази даних зображень з обличчями людей для розпізнавання. Кожен аспект або частина досліджуваного зображення розбивається на ознаки, і ці вектори використовуються для визначення чи є фрагмент зображення обличчям, чи ні.

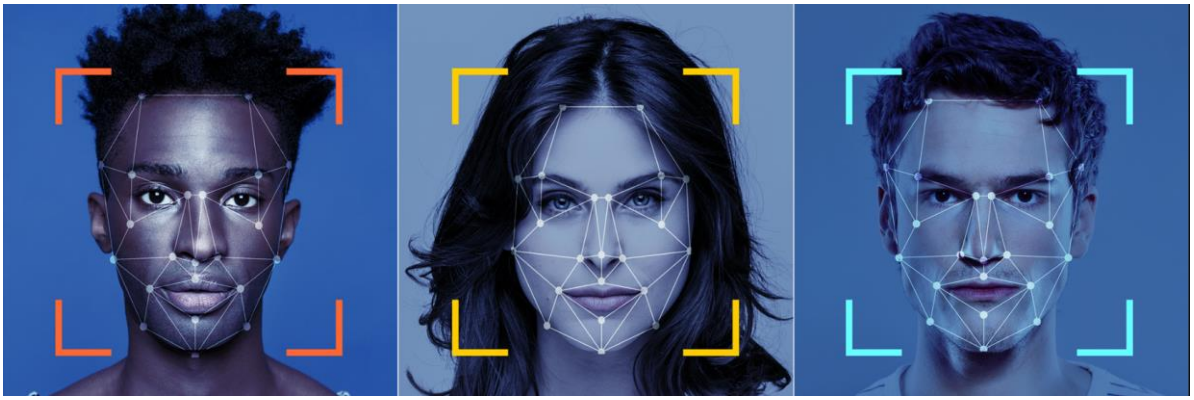


Рисунок 1.10 – Візуалізація емпіричного методу розпізнавання обличчя

З точки зору архітектури, можуть існувати суттєві розбіжності в різних системах, які розпізнають осіб, але принципово їхня робота базується на схожих принципах. Хоча існують різні методи розпізнавання обличчя, можна спостерігати спільну структуру процесу, як показано на рисунку 1.11.

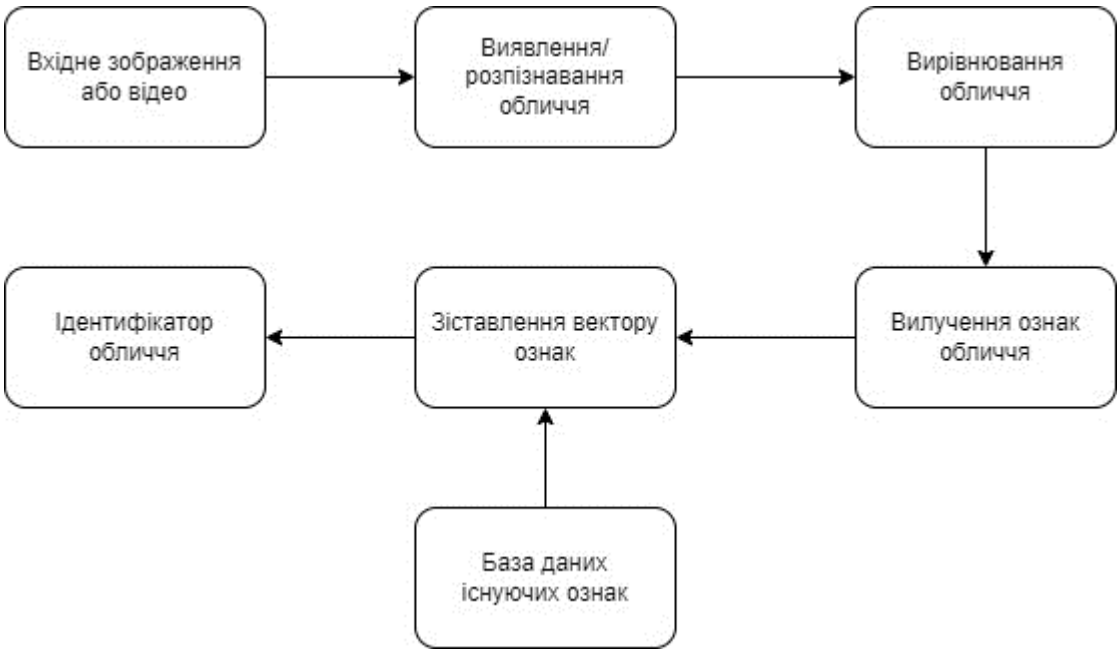


Рисунок 1.11 – Узагальнена архітектура процесу розпізнавання обличчя

Цей процес включає численні кроки, починаючи від виявлення та вирівнювання облич до аналізу даних зображення та зіставлення їх із відомими профілями в базі даних. Незважаючи на відмінності в реалізації, основний процес розпізнавання обличчя залишається відносно послідовним.

1.3.1 Приховані моделі Маркова

Одним із методів, що використовуються для розпізнавання обличчя, є приховані моделі Маркова (ПММ) з дискретним часом. ПММ використовують статистичні характеристики сигналів і враховують їх просторові властивості. Компоненти моделі складаються з групи прихованих станів, групи спостережуваних станів, матриці перехідних ймовірностей і початкової

ймовірності станів. Кожен з цих елементів має свою унікальну марковську модель. Під час розпізнавання об’єктів оцінюються моделі Маркова, створені для певного набору об’єктів, і шукається найвища спостережувана ймовірність того, що послідовність спостереження для конкретного об’єкта створена відповідною моделлю. Цей імовірнісний підхід до розпізнавання є важливим, оскільки він враховує як статистичні, так і просторові характеристики.

Щоб ефективно виконати задачу розпізнавання обличчя, необхідно виконати два основні етапи: виявлення обличчя на даній фотографії та подальша ідентифікація особи. Основні компоненти цього процесу включають використання серії спостережуваних символів, які в даному випадку відносяться до коефіцієнтів, згенерованих шляхом дискретного косинусного перетворення обличчя суб’єкта в межах попередньо визначеного вікна сканування. Інші ключові компоненти включають різноманітний діапазон станів, а також початковий вектор станів і ймовірностей, призначених як спостережуваним символам, так і переходам станів.

Використання ймовірнісної марковської моделі є дуже ефективним підходом до завдання розпізнавання обличчя завдяки його здатності враховувати низку важливих факторів. По-перше, поняття незалежності між спостереженнями забезпечує більшу легкість і точність у виявленні шаблонів у псевдовипадкових фрагментах обличчя. Крім того, кожна випадкова змінна забезпечує розподіл дискретних ймовірностей, які внутрішньо пов’язані з пов’язаними станами та всеохоплюючими моделями. Загалом цей підхід забезпечує ефективний і точний метод успішного розпізнавання обличчя.

1.3.2 Метод опорних векторів

Метод опорних векторів, також відомий як машини опорних векторів (SVM) – це алгоритм класифікації, який зазвичай використовується для класифікації даних на дві категорії, але існують варіанти методу, які можна використовувати для багатокласової класифікації. Підхід SVM базується на

концепції функції ядра, яка дозволяє трансформувати набір точок із простору з вимірами N_d у простір більшої вимірності з вимірами N_{KS}, відомий як простір ядра. У цьому просторі ядра можна знайти лінійний класифікатор, який може розділити два класи, навіть якщо точки даних не розподілені лінійно у вихідному просторі. Цей підхід корисний для вирішення великомасштабних проблем класифікації, оскільки SVM надає можливість пристосовуватися до складних зв'язків більш ефективним способом, ніж інші методи. На рисунку 1.12 представлений вигляд точок у двовимірному та у просторі ядра відповідно.

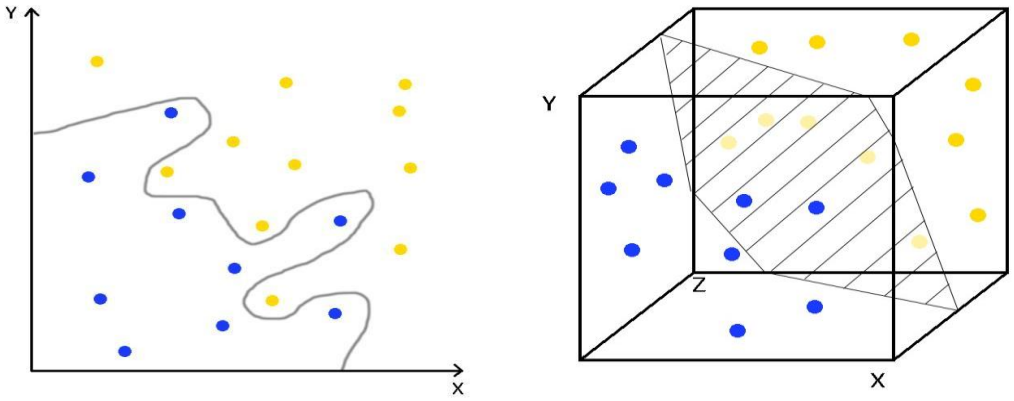


Рисунок 1.12 – Візуалізація точок у двовимірному просторі та просторі ядра

Класифікатор SVM вважається моделлю, яка пропонує максимальну прозорість, оскільки він вибирає гіперплощину в просторі ядра, яка не тільки розділяє два класи, але також максимізує відстань між представниками класів, які є найближчими до гіперплощини. Ці представники називають опорними векторами. Вони визначають положення гіперплощини та відіграють вирішальну роль, оскільки лише вони повинні зберігатися для прийняття рішень щодо класифікації майбутніх прикладів. Алгоритм SVM використовує гіперплощину для прийняття рішень, яка визначається наступним рівнянням:

$$\vec{\omega} * \vec{x} + b = 0$$

де x представляє точку в просторі ядра, а $\vec{\omega}$ – вектор, перпендикулярний до гіперплощини.

Вектор $\vec{\omega}$, який діє як визначальний фактор вирішальної гіперплощини в класифікаторі SVM є лінійною комбінацією опорних векторів. Подібним чином зсув b, який служить зсувом до гіперплощини, може бути визначений виключно за опорними векторами. Коли ці два параметри відомі, класифікатор SVM може правильно класифікувати нову точку x на основі її положення в одній із чотирьох областей простору ознак, визначеного гіперплощиною: глибоко в межах першого класу ($\vec{\omega} * x + b \geq 1$), поблизу межі між першим і другим класами ($\vec{\omega} * x + b > 0$), поблизу межі між другим і першим класами ($\vec{\omega} * x + b < 0$), або глибоко всередині другого класу ($\vec{\omega} * x + b \leq -1$).

Однією з ключових переваг SVM є те, що він добре працює в просторах великої розмірності, використовуючи гіперплощину для розділення різних класів даних. SVM передбачає пошук оптимальної гіперплощини, яка максимізує запас між двома класами. Функція ядра, яка є невід'ємною частиною SVM, допомагає перетворювати вхідні дані з низького розміру на високий. Трюк ядра не потребує явного обчислення функції відображення $\vec{\varphi}(x)$, оскільки всі обчислення в просторі ядра складаються лише зі скалярних векторних добутків. Ядро, яке використовується, сильно впливає на ефективність алгоритму з точки зору пошуку оптимальної гіперплощини для набору даних. SVM має численні програми, включаючи методи розпізнавання об'єктів, такі як HOG-SVM, які використовують гістограми орієнтованих градієнтів (HOG) як вектор ознак, що передається до класифікатора SVM для ідентифікації об'єктів на зображеннях за допомогою техніки ковзного вікна.

1.4 Постановка задачі розробки системи розпізнавання втомленості

Актуальність програмної системи виявлення сонливості на основі OpenCV полягає в тому, що безпека на дорозі є надзвичайно важливою і сонливість водіїв є серйозною проблемою, що може призвести до аварій. Основні причини сонливості під час керування автомобілем включають довгі години руху, недостатній сон, фізичне та психологічне виснаження. Розробка програмної

системи, здатної виявляти сонливість водіїв, має велике значення для підвищення безпеки на дорозі. Використання технологій комп'ютерного зору та нейронних мереж на основі OpenCV дозволяє точно визначати ознаки сонливості, такі як позіхання та зниження руху очей. Така система може бути використана в різних типах транспортних засобів, включаючи автомобілі, автобуси та вантажівки. Вона може бути особливо корисною для довгих відрядних поїздок, де ризик сонливості збільшується. Отже, актуальність зумовлена недосконалістю сучасних технологій детектування стану оцей водія для уникнення аварійних ситуацій з використанням машинного навчання. Далі, наведено основні функціональні вимоги та можливості системи:

1. Виявлення ознак втомленості: Система повинна бути здатна розпізнавати фізичні та поведінкові ознаки втомленості, такі як миттєве засипання, повільна реакція, мимовільні рухи, зниження концентрації тощо.

2. Аудіовізуальні сигнали: Система може надавати аудіовізуальні сигнали водію для сповіщення про його втомленість, такі як звукові сигнали, сповіщення на панелі приладів або вібрація крісла.

3. Система рекомендацій: Залежно від виявленого рівня втомленості, система може надавати рекомендації водію, наприклад, зробити перерву на відпочинок, зупинитися на заправці, виконати фізичні вправи.

4. Інтеграція з іншими системами автомобіля: Система може бути підключена до інших систем автомобіля, таких як система адаптивного круїз-контролю, система попередження про зіткнення або система управління стабільністю, для покращення безпеки та забезпечення відповідних заходів.

Висновки до розділу 1

У першому розділі проведено аналіз предметної області, визначено основні аналогічні системи, систематизовані переваги і недоліки кожної з них. Проведено постановку задачі на розробку програмної системи розпізнавання сонливості водія, визначено функціональні та нефункціональні вимоги.

2. ОГЛЯД ІСНУЮЧИХ АЛГОРИТМІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ РІВНЯ ВТОМЛЕНОСТІ ВОДІЯ

2.1 Алгоритм головних компонент

Алгоритм головних компонентів (PCA) – це статистичний метод, який використовує перетворення Карунена-Лоева для зменшення складності вхідних даних шляхом стиснення їх у менші інформативні характеристики. Спочатку він був розроблений, щоб мінімізувати розмірність простору ознак, зберігаючи важливі характеристики. У розпізнаванні сонливості водія PCA широко застосовується для представлення зображення людини за допомогою вектора менших розмірів, відомих як головні компоненти, проти еталонних векторів, що зберігаються в базі даних. Основною метою цього методу є видалення зайвих функцій даних, зберігаючи суттєву та значущу інформацію, необхідну для подальшого аналізу. Він працює шляхом аналізу набору зображень обличчя та визначення варіацій серед них. Потім ці варіації можна описати набором ортогональних векторів, які називаються власними гранями. Метою методу головних компонентів є виділення найбільш релевантних характеристик із зображень, які можуть бути корисними для таких завдань, як розпізнавання та маніпулювання зображеннями. Його основна мета – зменшити розмірність простору ознак, зберігаючи при цьому варіативність, присутню в навчальній вибірці, що робить його цінним інструментом для розпізнавання сонливості водія за кермом та інших пов’язаних додатків.

У технології розпізнавання обличчя власні вектори – це набір математичних векторів, які є вирішальними для кодування інших зображень обличчя за допомогою комбінації зважених значень. Під час фази навчання системи розпізнавання обличчя ці власні вектори обчислюються з набору зображень обличчя та використовуються для створення зображення обличчя людини. Зображення можна далі перетворити на вектор коефіцієнтів і зберегти в базі даних як запис обличчя, який служить ключем для ідентифікації людини.

Таким чином, лише вибрана кількість власних векторів використовується для кодування зображення обличчя людини, що робить можливим зберігати та отримувати інформацію про ідентифікацію людини швидко та ефективно.

На рисунку 2.1 представлено принцип роботи алгоритму розпізнавання облиичя з використанням алгоритму головних компонент (PCA).

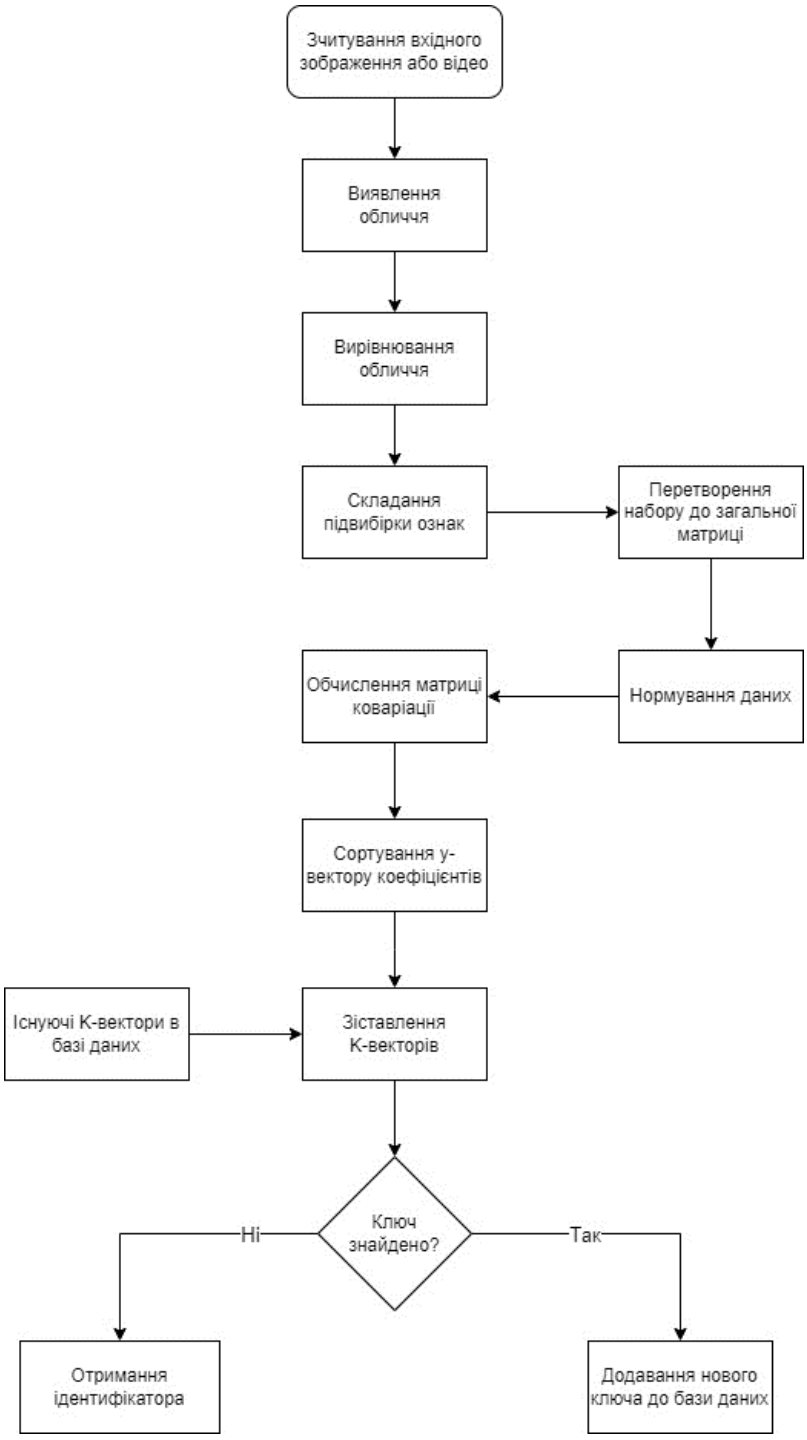


Рисунок 2.1 – Узагальнена блок-схема алгоритму головних компонент

Метод головних компонентів використовується для перетворення набору зображень індивідів у загальну матрицю даних у декілька етапів. Цей процес передбачає розміщення зображення кожної людини в рядку в матриці після зміни розміру до однакового розміру та нормалізації їхніх гістограм. Однак ефективність методу головного компонента може бути обмежена в сценаріях, коли на зображенні існують значні варіації в освітленні або виразі обличчя. Це пояснюється тим, що методика покладається на апроксимацію вхідного набору даних, а не на розрізнення різних класів осіб. Це означає, що метод може бути нездатним точно диференціювати людей за наявності великих варіацій у вхідних даних. Незважаючи на свої обмеження, метод широко використовується на практиці виявленні втомленості водія за кермом.

2.2 Алгоритми з використанням нейронних мереж

В даний час доступні різні типи нейронних мереж, і одним із найбільш часто використовуваних варіантів є багатошарова персептронна мережа, яка може класифікувати дане вхідне зображення або сигнал на основі його попередньо налаштованих параметрів. Нейронні мережі вчаться з набору навчальних прикладів, а процес навчання включає коригування зв’язків між нейронами за допомогою методу градієнтного спуску для вирішення проблеми оптимізації. У процесі навчання нейронних мереж автоматично виділяються ключові ознаки, визначається їх релевантність і встановлюються їхні зв’язки. Завдяки цьому нейронна мережа отримує можливість класифікувати вхідний сигнал і легко виділяти важливі характеристики. Популярною архітектурою штучної нейронної мережі, яка широко використовується в програмах комп’ютерного зору, є згорткова нейронна мережа (CNN). Ця архітектура містить нейрони, які спеціально розроблені для виявлення певних візуальних характеристик, таких як вертикальні, горизонтальні чи діагональні межі. Ці нейрони організовані у стержнеподібну структуру та об’єднуються для отримання остаточного візуального сприйняття. Алгоритм CNN імітує цю

структуру та взаємодію нейронів, дозволяючи мережі точно ідентифікувати та класифікувати конкретні особливості зображення. Аналізуючи необроблені піксельні дані та створюючи ієрархічні уявлення, модель CNN може розпізнати риси зображення та класифікувати його за відповідними мітками.

Згорткова нейронна мережа – це тип алгоритму глибокого навчання, який приймає цифрові матриці як вхідні дані. Вхідними даними може бути зображення в градаціях сірого, представлене однією матрицею, або багатошарове зображення RGB, представлене трьома матрицями.

Мережа містить фільтр, який має попередньо визначений набір значень, який допомагає ідентифікувати конкретні характеристики або особливості вхідного зображення. Розробка ефективної архітектури для розпізнавання активності за допомогою згорткової нейронної мережі потребує оцінки складності шарів обробки, кількості шарів і швидкості виведення.

Кількість шарів у згортковій нейронній мережі змінюється залежно від її конструкції та проблеми, що розглядається, причому перший шар завжди є згортковим, що дає назву мережі. Вихідна матриця створюється після того, як операція згортки виконується як на матриці фільтра, так і на частині матриці зображення. Отриманий розмір матриці можна порівняти з розміром матриці фільтра. Кожен елемент результуючої матриці обчислюється шляхом множення матриці фільтра на підматрицю такого ж розміру (частина зображення) за допомогою формули скалярного добутку (2.1):

$$C_{i,j} = \sum_{u=0}^{m_x-1} \sum_{v=0}^{m_y-1} A_{i+u,j+v} B_{u,v'} \qquad (2.1)$$

де A, B – матриці з розмірністю $n_x \times n_y$ та $m_x \times m_y$ відповідно, C – вихідна матриця з розмірністю $(n_x - m_x + 1) \times (n_y - m_y + 1)$.

У згорткових нейронних мережах, коли зображення проходить через більшу кількість згорткових шарів і субдискретизується, карти активації стають більш складними, визначаючи складні характеристики. Шари підвибірки призначені для зменшення розмірності карт активації попереднього згорткового шару. В першу чергу це робиться для того, щоб виключити непотрібні деталі, які

вже були оброблені в попередньому шарі. Крім того, це допомагає уникнути перетренованості, фільтруючи зайву інформацію. Зменшуючи розмірність карт, рівень підвибірки допомагає зробити нейронну мережу обчислювально недорогою, зберігаючи релевантну інформацію. У сфері архітектури нейронної мережі рівень підвибірки відповідає за зменшення масштабу карти функцій згорткової нейронної мережі. На відміну від згорткового рівня, рівень підвибірки використовує для цього процесу ядро, що не перекривається.

Як правило, розмір ядра, що використовується в шарі підвибірки, становить половину розміру ядра, що використовується в попередньому рівні. Карту ознак розділено на менші комірки, і максимальне значення з кожної комірки вибирається як результат шару підвибірки. Крім того, функція активації випрямленої лінійної одиниці (ReLU) часто використовується на рівні субдискретизації, яка змінює вихідне зображення від операції згортки, щоб полегшити більш ефективну оцінку нейронною мережею. Рівень підвибірки відіграє важливу роль у вилученні ознак у нейронній мережі.

У згортковій нейронній мережі (CNN) повністю зв’язаний рівень є останнім типом шару і зазвичай реалізується як багатошаровий перцептрон. Його основна роль полягає у виконанні класифікації шляхом моделювання складних нелінійних функцій та їх оптимізації для підвищення якості розпізнавання системи. Архітектура CNN передбачає багаторазове згортання вхідного зображення з подальшим ущільненням за допомогою підвибірки, що призводить до формування більш абстрактних карт ознак. Коли зображення проходить через наступні шари, кількість каналів зазвичай збільшується, а розмірність зображення зменшується в кожному каналі. Завдяки цьому процесу створюється набір із кількох каналів, які інтерпретуються як абстрактні концепції, які можна знайти на вихідному зображенні. Повністю підключений рівень відповідає за класифікацію цих складних нелінійних функцій і покращує якість розпізнавання системи. Результатом цього процесу є набір абстрактних карт функцій, які можна використовувати для інтерпретації зображень. Передача даних у нейронних мережах досягається за допомогою процесу об’єднання даних

і передачі їх у повнофункціональну нейронну мережу. Нейронна мережа може мати різні рівні, що дозволяє більш складний і складний аналіз даних. Незважаючи на те, що цей тип нейронної мережі є ефективним, з'єднання шарів призводить до втрати просторової структури в пікселях і, отже, сітка стає меншою за вихідне зображення.

Щоб краще зрозуміти цю концепцію, на рисунку 2.2 наведено візуалізацію нейронної мережі, яка включає багатошаровий перцептрон.

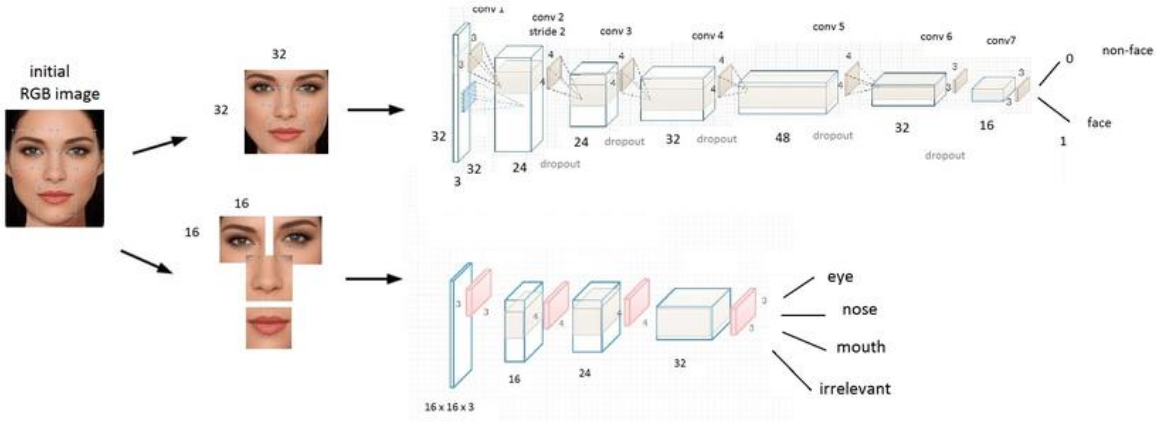


Рисунок 2.2 – Візуалізація роботи згорткової нейронної мережі для задачі розпізавання обличчя та подальшої класифікації

Вихідний рівень нейронної мережі є ключовим компонентом, який визначає ймовірність належності даного вхідного зображення до певної категорії. Щоб досягти цього, кількість нейронів у вихідному шарі має відповідати кількості класів ознак у наборі даних. Потім до зваженого та підсумованого сигналу, створеного вихідним шаром, застосовується функція активації для подальшого уточнення результатів.

Для навчання моделі використовується добре відомий і широко використовуваний метод, який називається зворотним поширенням помилок. У цьому методі нейронна мережа вчиться на своїх помилках і відповідно коригує свої ваги, щоб покращити продуктивність вихідного рівня.

2.3 Алгоритм Віоли-Джонса

Алгоритм Віоли-Джонса є відомим рішенням для розпізнавання втоми водія, оскільки він ефективний у ідентифікації об’єктів на зображеннях. Цей метод вперше був розроблений Полом Віолою та Майклом Джонсом у 2001 році. Він використовує техніку ковзного вікна, яка створює рамку, яка регулюється, щоб охопити основні характеристики зображення, і дотримується каскадного протоколу Хаара як ключових функцій. Після завершення процесу навчання вхідні зображення класифікуються за допомогою каскадного алгоритму Хаара. Цей алгоритм розділяє вхідні дані на дві категорії залежно від наявності або відсутності визначеного об’єкта. Метод Віоли-Джонса вважається високоефективним у розпізнаванні об'єктів і має кращу якість та швидкість обробки в порівнянні з іншими сучасними методами. Крім того, у цьому методі ймовірність хибнопозитивних результатів під час процесу класифікації значно нижча. Перед застосуванням будь-яких методів розпізнавання обличчя алгоритм перетворює вхідне зображення в градації сірого, що допомагає зменшити обсяг даних.

Перший етап передбачає перетворення кольорового зображення, що складається з трьох кольорових каналів (червоного, зеленого та синього), у зображення у градаціях сірого. Цей процес зменшує яскравість зображення до 256 рівнів, які потім перетворюються на матрицю яскравості, що дає змогу обчислити загальну яскравість будь-якої прямокутної області зображення. Ця матриця служить інтегрованим представленням зображення в градаціях сірого та скорочує час обчислення, необхідний для визначення яскравості конкретної області, незалежно від її розміру. Отримана матриця відображає розміри вихідного зображення та створюється шляхом додавання сукупної яскравості кожного пікселя над і ліворуч від кожного елемента матриці (2.2):

$$L(x,y) = \sum_{i=0,j=0}^{i\le x,j\le y} I(i,j) \qquad (2.2)$$

де $I(i,j)$ – яскравість пікселів вихідного зображення.

Матриця $L(x, y)$ – це набір елементів, що представляють суму пікселів всередині прямокутника, починаючи з верхнього лівого кута (0, 0) і закінчуючи позицією (x, y). Значення кожного пікселя (x, y) визначається складанням значень усіх пікселів, розташованих ліворуч і над ним. Для обчислення повного зображення потрібен лінійний час, який прямо пропорційний кількості пікселів у зображенні, що дозволяє отримати все зображення за один прохід. Другий крок у процесі передбачає розміщення маски Хаара, яка є квадратною маскою з набором прямокутних областей, на матрицю яскравості, отриману в результаті попереднього кроку. Ці прямокутні області прилягають одна до одної та розділені на дві групи з різною висотою та шириною, розташовані в різних місцях зображення. Маска систематично рухається, змінюючи своє положення та розмір, охоплюючи таким чином різні області зображення в різний час.

Ознака f – це відображення $f: X \to D_f$, де X містить множину досліджуваних об'єктів, а D_f – множина, яка містить допустимі значення ознаки f. У випадку наявності ознак $f_1, \dots, f_n$, вектор ознак $x = (f_1(x), \dots, f_n(x))$ ідентифікується, як ознаковий опис об'єкта $x \in X$. На рисунку 2.3 представлений приклад навчання класифікатора Хаара.

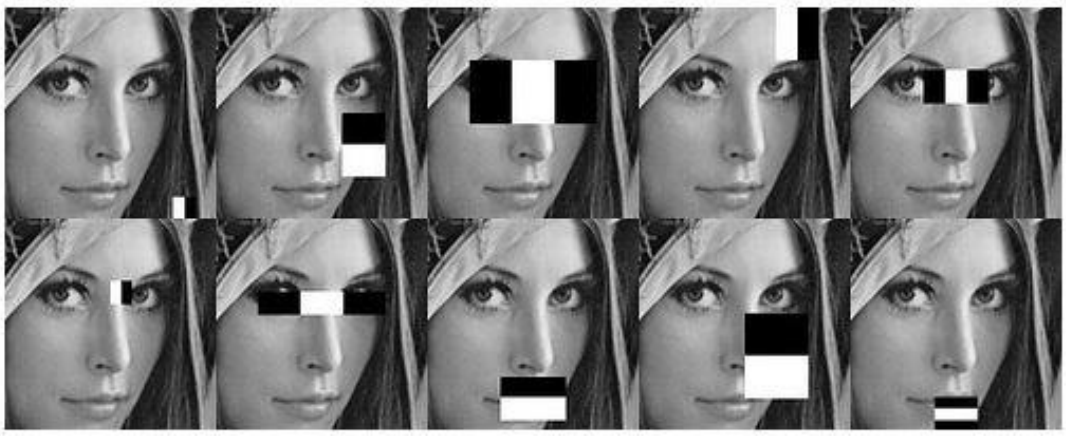


Рисунок 2.3 – Візуалізація навчання класифікатора на основні каскадів Хаара для вилучення ознак обличчя на зображеннях [18]

2.4 Детектор ознак FAST

Дектор ознак FAST (Features from Accelerated Segments Test) спочатку був запропонований Ростеном і Драммондом. Він включає в себе порівняння центральної точки (P) із набором точок у межах малого кола, що її оточує. Цей метод спрямований на те, щоб визначити, чи P є хорошою сингулярною точкою, аналізуючи, наскільки близькі точки подібні до P. Алгоритм SUSAN (найменший однозначний сегмент асиміляції ядра), рання реалізація цієї концепції, порівнював усі точки кола з центром P. Однак алгоритм FAST, який можна розглядати як вдосконалення SUSAN, вніс два важливі покращення в цей підхід. По-перше, він використовує лише точки, знайдені на конкретному контурі, визначеному алгоритмом Бразенхема, замість порівняння з усіма точками на колі. По-друге, окремі контурні точки класифікуються за однією з трьох категорій: темніші за P, світліші за P або подібні до P. Ця класифікація базується на пороговому значенні (t), де розглядаються пікселі з яскравістю, нижчою за $I_{\text{p}} - t$ темні, а пікселі з яскравістю, більшою за $I_{\text{p}} + t$ є світлими. Щоб підвищити швидкість виявлення спеціальних точок, алгоритм FAST використовує додаткову стратегію оптимізації перед класифікацією.

Зокрема, лише чотири точки, розташовані на однаковій відстані від точки P, перевіряються перед класифікацією. Емпірично було встановлено, що якщо принаймні дві з цих точок не яскравіші або темніші за P, то P не можна вважати ознакою FAST. Це призводить до значного зменшення часу пошуку по всьому зображенню, як показано на рисунку 2.4. Для полегшення розуміння пікселі, які перевіряються під час класифікації, пронумеровані, причому пікселі 1, 5, 9 і 13 перевіряються для проведення початкової оцінки точки P.

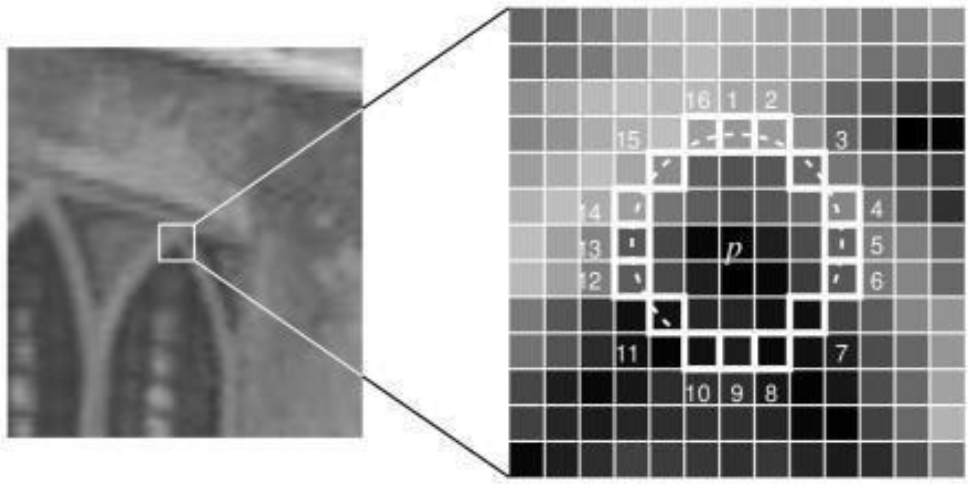


Рисунок 2.4 – Приклад вилучення ознак з використанням алгоритму FAST

У цього методу є недолік, оскільки він схильний помилково розпізнавати кілька сусідніх пікселів як кути. Щоб вирішити цю проблему, алгоритм FAST надає оцінку для кожного кута та відфільтровує будь-які особливі точки, які є суміжними з особливою точкою з вищим показником. Розрахунок балів передбачає підсумовування абсолютних різниць між центральним пікселем і його світлішими й темнішими сусідніми пікселями. Потім вибирається максимальний бал. Ця техніка допомагає краще виявляти справжні кути зображення, мінімізуючи кількість помилкових спрацьовувань:

$$score = \max\left(\sum_{X\in\{brighter\}}\left|I_x - I_p\right| - t, \sum_{X\in\{darker\}}\left|I_x - I_p\right| - t\right) \quad (2.3)$$

2.5 Детектор ознак SIFT

Алгоритм SIFT (Scale Invariant Feature Transform), який був розроблений Девідом Лоуі, сьогодні широко використовується, оскільки він є основою для кількох інших функцій. Незважаючи на те, що обчислення функцій SIFT вимагає значних обчислювальних ресурсів, вони мають високу експресивність і, отже, ідеальні для таких завдань, як розпізнавання та відстеження. Як показано на

рисунку 2.5, алгоритм SIFT є ефективним механізмом для ідентифікації ключових точок у цифрових зображеннях, незмінних щодо масштабу. Незважаючи на високі обчислювальні вимоги, функції SIFT здатні точно виявляти та відстежувати об’єкти на різних зображеннях.

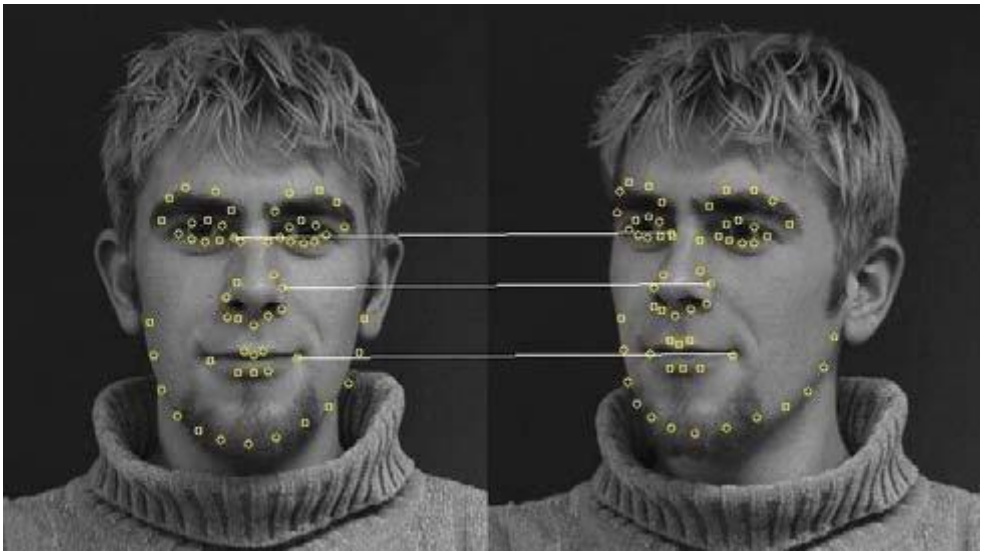


Рисунок 2.5 – Результат роботи детектора ознак SIFT

Метод SIFT відомий своєю властивістю інваріантності до масштабу, яка досягається на початковому етапі алгоритму. На цьому етапі вихідне зображення проходить серію згорток із гауссовими ядрами різного розміру, в результаті чого утворюється масштабований простір зображення, який представлений на рисунку 2.6. Звідси генерується піраміда Гауса шляхом поділу масштабованого простору на секції або октави, причому кожна наступна октава зменшує розмір зображення вдвічі. Одночасно піраміда різниці Гауса (DoG) будується із сусідніх зображень у піраміді Гауса, яка є різницею між згладженими зображеннями Гауса. Потім ці набори зображень порівнюються піксель за пікселем, де кожен піксель порівнюється не лише з вісьмома сусідами на тому самому зображенні, але й з 18 пікселями в попередньому та наступних зображеннях у стеку. Якщо значення різниці Гаусса для пікселя вище або менше, ніж набір порівняння з 26 пікселів, то це вважається екстремумом масштабованого простору DoG.

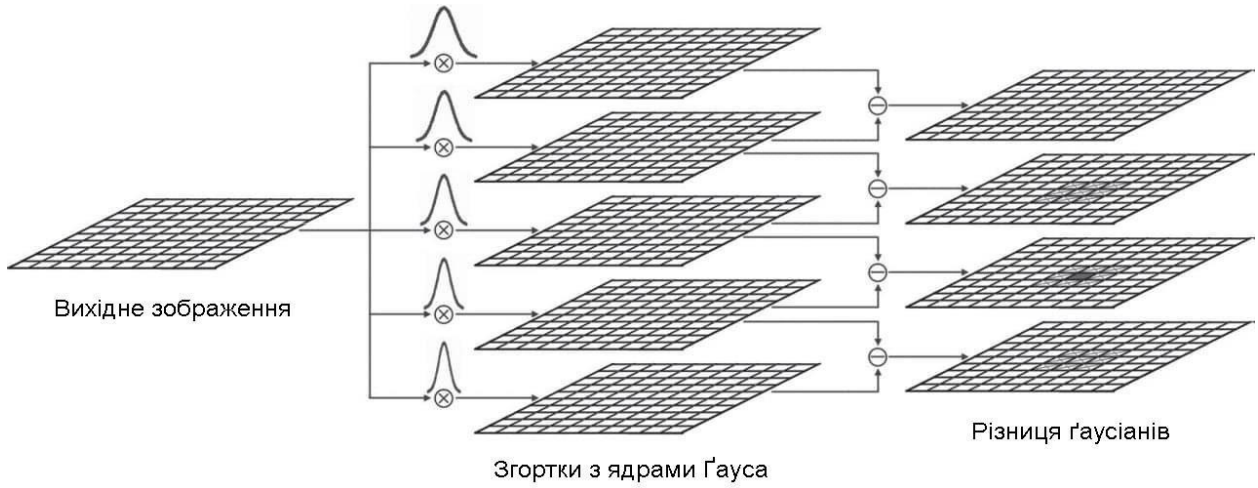


Рисунок 2.6 – Схема побудови згортки DoG

Алгоритм SIFT визначає екстремуми в масштабованому просторі та продовжує створення дескрипторів для кожного об’єкта. Для побудови дескрипторів спочатку визначається напрям спеціальної точки шляхом порівняння похідних за напрямком навколо неї та вибору напрямку з найвищою похідною. Після визначення основного напрямку на його основі обчислюються інші властивості дескриптора. Це робить функції SIFT інваріантними не лише щодо масштабу, але й щодо напрямку. Після визначення масштабу та орієнтації локальний дескриптор зображення обчислюється за допомогою локальних градієнтів. Коло навколо спеціальної точки обертається відповідно до напрямку дескриптора, а пікселі групуються в області, як правило, квадрат 4×4 із 16 пікселями навколо певної точки.

Для кожного регіону будується гістограма напрямків у всіх його точках, що складається з восьми інтервалів. Зазвичай існує шістнадцять регіонів, які разом створюють 128 компонентів. Цей 128-компонентний вектор служить дескриптором певної точки в алгоритмі SIFT. Висока описова здатність функцій SIFT є результатом такої великої кількості компонентів. Саме такий високий рівень деталізації та специфічності робить функції SIFT популярним вибором для обробки та розпізнавання облич на зображеннях.

2.6 Дескриптор BRISK

Дескриптор функції BRISK спочатку був представлений як вдосконалення широко використовуваного дескриптора BRIEF. Основною метою цього вдосконалення була розробка більш стабільної та надійної версії дескриптора. Дескриптор BRISK має спеціальний детектор точок, який має низку переваг перед дескриптором BRIEF. BRISK, детектор ознак, який використовується в обробці зображень, заснований на FAST-подібному детекторі AGAST. Однак BRISK покращує це, не лише виявляючи елементи, але й визначаючи їхній масштаб та орієнтацію.

Щоб визначити масштаб функції, BRISK створює піраміду в масштабованому просторі з фіксованою кількістю шкал, де кожна сусідня шкала відрізняється у два рази, а потім обчислює фіксовану кількість інтраоктав на шкалу. Першим кроком детектора ознак BRISK є застосування алгоритму FAST для пошуку ознак на всіх рівнях піраміди. Потім видаляються немаксимальні характеристики, тобто ті, у яких показник не є найбільшим серед усіх сусідів. Після побудови списку функцій BRISK обчислює показник FAST у відповідних точках на попередньому та наступному зображеннях масштабу. Оцінки AGAST апроксимуються параболою, яка потім використовується для визначення максимального значення, що вказує справжній масштаб функції BRISK.

Як наслідок, масштаб функції не обов’язково може бути пов’язаний з одним із зображень у піраміді. Подібна інтерполяція застосовується до піксельних координат, тому об’єкти відображаються на субпіксельних координатах. На рисунку 2.7 представлена архітектура дескриптора BRISK.

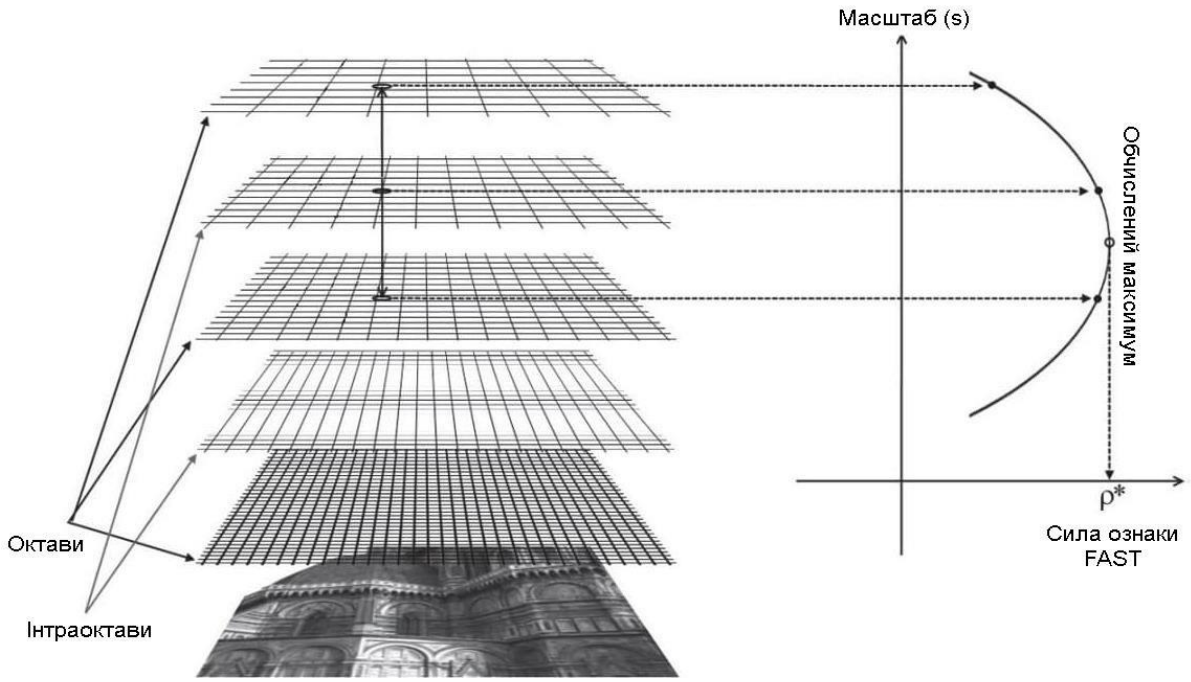


Рисунок 2.7 – Архітектура дескриптора BRISK

Щоб збільшити масштаб, функції BRISK також враховують орієнтацію. Побудова дескриптора BRISK включає кілька контурів, що оточують центральну точку. Кожному контуру присвоюється Ki обраних точок, яким, у свою чергу, присвоюється кругова ділянка. Діаметр кругової ділянки дорівнює довжині кола C_i, що містить обрану точку, поділеній на K_i. Кругла ділянка відповідає зразку, який згорнуто з ядром Гауса радіуса σi = $C_i/2K_i$.

У процесі генерації бітового дескриптора виконується порівняння яскравості між парами точок у всіх колах. Це робиться шляхом поділу пар на дві підмножини: близькі та далекі. Близькі пари – це ті пари точок, відстань яких менша за задане порогове значення (dmax), тоді як віддалені пари – це ті, відстань яких перевищує задане порогове значення (dmin). Розраховуючи різницю яскравості між парами точок і нормалізуючи їх на відстань між точками, отримують локальний градієнт. Це робиться для кожної пари віддалених точок. Ці локальні градієнти підсумовуються, після чого з цих сум виводиться домінуючий напрямок дескриптора. Дескриптор, отриманий бінаризацією градієнтів, обчислених за близькими парами, не залежить від орієнтації. Це

робиться шляхом обчислення ознак, визначених близькими парами відносно напрямку, отриманого на попередньому етапі. Регулюючи значення dmax можна отримати дескриптор довільної довжини.

2.7 Детектор ознак SURF

SURF (Speeded-Up Robust Features) – це алгоритм, який базується на алгоритмі SIFT (Scale-Invariant Feature Transform). Алгоритм SURF було розроблено з конкретною метою покращення обчислювальної ефективності алгоритму SIFT шляхом заміни певних частин більш ефективними, зберігаючи при цьому високий рівень точності для проблем розпізнавання. Це призводить до швидшого виявлення функцій, а спрощений алгоритм означає, що функції більш стійкі до змін орієнтації та освітлення порівняно з SIFT. У цьому підході все зображення трансформується один раз, що дозволяє обчислити загальну яскравість у будь-якій прямокутній області за допомогою простої арифметичної операції. Подібно до інших детекторів ознак, SURF визначає унікальну точку на зображенні на основі локального гессе, де форма Гессе визначається матрицею, яка характеризує кривизну зображення та зміну країв:

$$\frac{d^2}{dx_i dx_j} G(\vec{x}, \sigma) \qquad (2.4)$$

де $G(\vec{x}, \sigma)$ – нормоване ядро Ґауса розміру σ.

Алгоритм SURF використовує прямокутні фільтри для виявлення ознак і, на відміну від SIFT, не потребує піраміди масштабованих зображень завдяки використанню цілісних зображень. Натомість для обчислення Гессе застосовують більші прямокутні фільтри в різних масштабах. Особливі точки, які вказують на локальні екстремуми визначника Гессе над певним порогом, визначаються як особливості в SURF. Для визначення напрямку цих особливостей використовується інтегральне зображення для оцінки

максимального локального градієнта. Два простих вейвлети Хаара застосовуються до різних частин області навколо сингулярної точки в масштабованому просторі для апроксимації локальних градієнтів, причому вейвлети розміром 4s і кроком s використовуються в межах кола радіусом 6s, якщо масштаб ознаки дорівнює s. Щоб створити дескриптор SURF для об'єкта, обчислюються градієнти у вікні та вибирається напрямок із максимальним результатом. Отриманий напрямок використовується для обчислення дескриптора вектора, що забезпечує інваріантність до обертання. Цей дескриптор обчислюється шляхом поділу прямокутника 20x20s на 16 квадратів у сітці 4x4, яка потім повертається відповідно до обчисленого напрямку об'єкта. У кожній клітинці сітки 25 пар вейвлетів Хаара використовуються для апроксимації градієнтів x і y зображення. Потім обчислюються суми та суми абсолютних значень 25 згорток із цими вейвлетами в напрямках x і y, утворюючи чотири числа для кожної клітинки сітки 4x4. Ці числа є компонентами 64-вимірного вектора, який використовується для дескриптора SURF спеціальної точки, як показано на рисунку 2.8. Цей підхід гарантує, що дескриптор не залежить від орієнтації зображення, і дозволяє витягувати надійні функції із зображень.

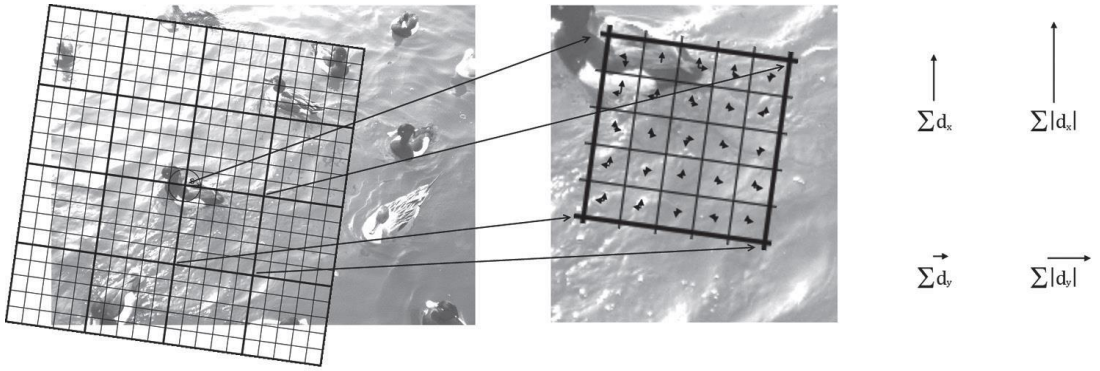


Рисунок 2.8 – Основні етапи обчислення дескриптора SURF: 1) накладання сітки, 2) розбиття комірки на підкомірки, для яких будуть розраховані похідні за напрямом, 3) відбувається підсумовування отриманих похідних, що в результаті дає чотири значення для кожної комірки

2.8 Розроблений алгоритм розпізнавання втомленості водія

Для побудови системи детектування рівня втомленості людини, необхідно враховувати наступні структурні елементи:

- Наявність доступу до камери.
- Алгоритм виявлення орієнтирних рис обличчя.
- Алгоритм визначення стану «закритих повік».

Розглянемо детальніше пункти для виявлення орієнтирних рис обличчя та визначення стану «закритих повік». Для визначення стану очей буде використана математична формула, відома як співвідношення сторін очей (EAR). Нижче представлена формула для проведення розрахунків (2.2).

$$EAR = \frac{||P_2 - P_6|| + ||P_3 - P_5||}{2||P_1 - P_4||} \quad (2.2)$$

Формула EAR повертає скалярну величину, яка відображає рівень відкриття очей. На рисунку 2.9 представлена візуалізація орієнтирів очей, а також графік співвідношення сторін очей людини з часом.

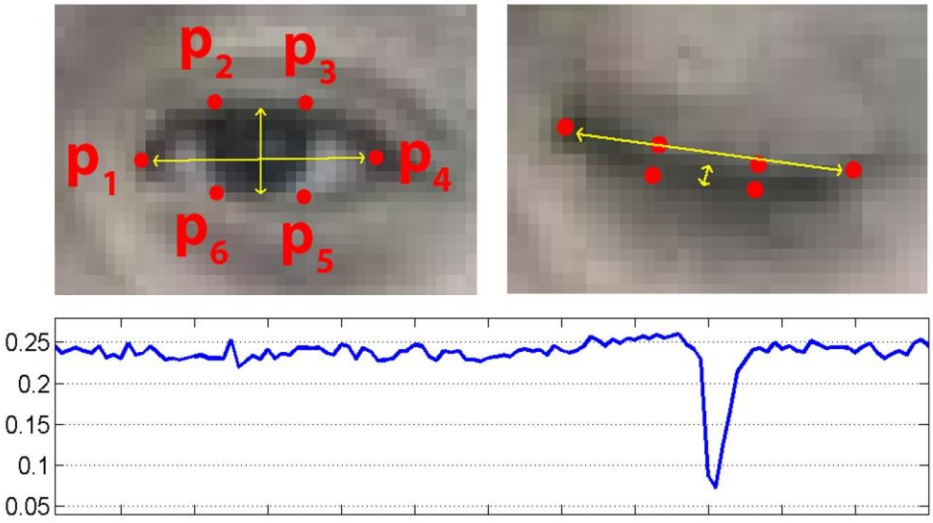


Рисунок 2.9 – Візуалізація відкритих та закритих очей людини [10]

Як видно з рисунку 2.9, коли око відкрите, співвідношення сторін ока, що повертається, має приблизно постійне позитивне значення. Однак, коли око моргає, значення співвідношення сторін раптово зменшується до нуля. Це означає, що співвідношення сторін ока, що повертається, є надійним параметром для визначення відкритості ока, особливо під час моргання.

На основі вищенаведених етапів виявлення сонливості людини сформировано блок-схему, яка представлена на рисунку 2.10.

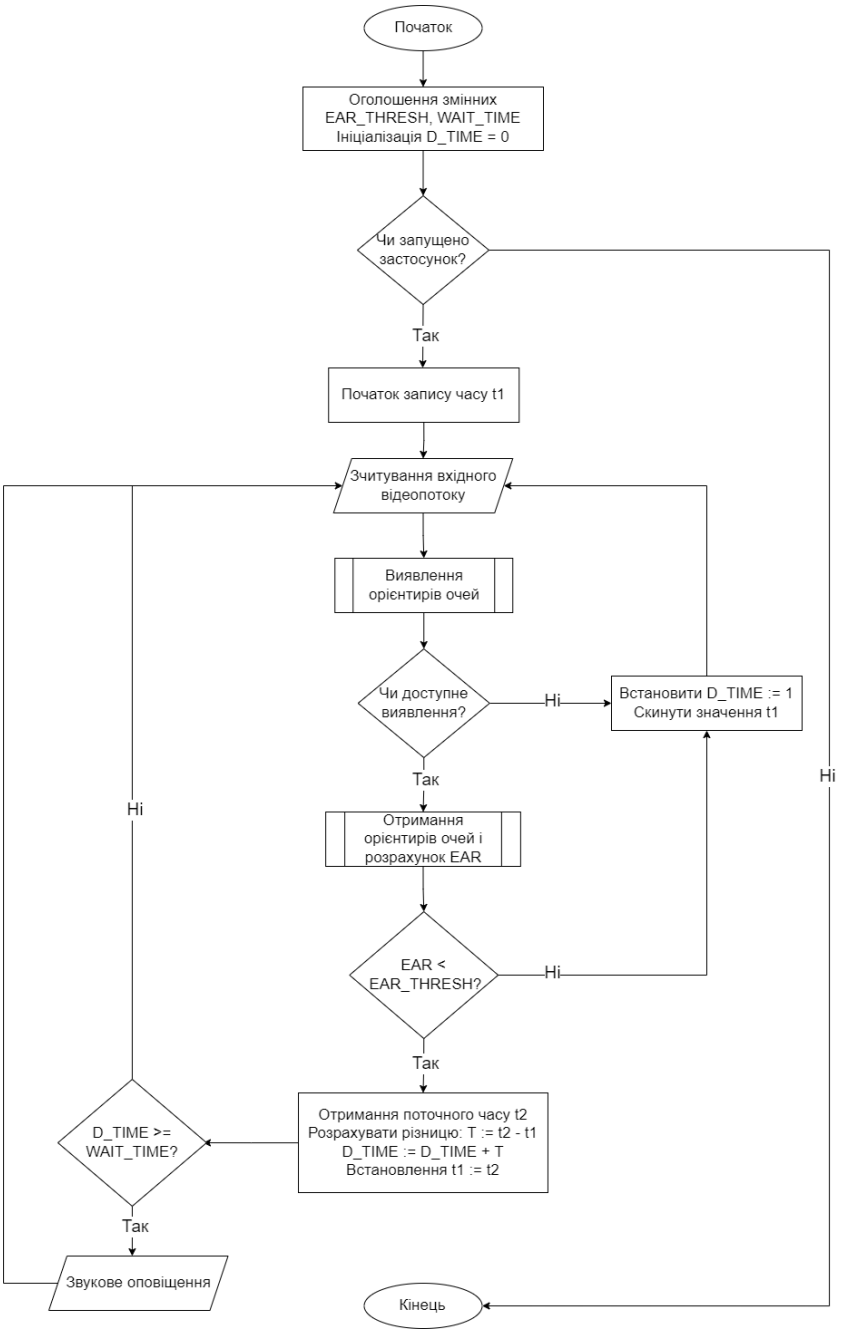


Рисунок 2.10 – Блок-схема алгоритму виявлення втомленості людини

Розглянемо детальніше етапи алгоритму виявлення сонливості:

- Оголошуються два порогових значення та лічильник: THRESH_EAR: порогове значення для перевірки, чи поточне значення EAR знаходиться в межах діапазону; D_TIME: змінна лічильника для відстеження часу, що минув із поточним EAR < THRESH_EAR; WAIT_TIME: щоб визначити, чи кількість часу, пройденого з EAR < THRESH_EAR, перевищує допустиму межу.
- Коли програма запущена, записується поточний час (у секундах) у змінну t1 і зчитується вхідний відеопоток з веб-камери.
- Проводиться попередня обробка даних з веб-камери з використанням пайплайну обробки даних для виділення контурів обличчя.
- Отримуємо відповідні (P_i) орієнтири для очей, якщо доступні виявлені орієнтири. В іншому випадку змінна t1 і D_TIME скидуються до початкових значень (D_TIME скидається тут, щоб зробити алгоритм послідовним).
- Якщо виявлення доступне, обчислюється середнє значення EAR для обох очей, використовуючи отримані орієнтири з попередніх кроків.
- Якщо поточний EAR < THRESH_EAR, додаємо різницю між поточним часом t2 і t1 до D_TIME.
- Якщо D_TIME >= WAIT_TIME, запускається звукове оповіщення про підвищений стан сонливості або відбувається перехід до наступних кадрів.

Висновки до розділу 2

В другому розділі проведено дослідження основних алгоритмів, які використовуються при розробці системи розпізнавання втомленості водія. Визначено, що метод головних компонентів використовується для перетворення набору зображень індивідів у загальну матрицю даних у декілька етапів. Цей процес передбачає розміщення зображення кожної людини в рядку в матриці після зміни розміру до однакового розміру та нормалізації їхніх гістограм. Однак ефективність методу головного компонента може бути обмежена в сценаріях, коли на зображенні існують значні варіації в освітленні або виразі обличчя. Це

пояснюється тим, що методика покладається на апроксимацію вхідного набору даних, а не на розрізнення різних класів осіб. Це означає, що метод може бути нездатним точно диференціювати людей за наявності великих варіацій у вхідних даних. Проаналізовані дескриптори та детектори SURF, BRISK, SIFT, FAST.

3. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ВИЯВЛЕННЯ ВТОМЛЕНОСТІ ВОДІЯ

3.1 Опис використаних технологій та мов програмування

Для розробки системи розпізнавання емоційного стану людини були використані наступні технології: Python, OpenCV, Keras, Tensorflow, MediaPipe, Torch. Розглянемо детальніше кожну з перерахованих технологій.

3.1.1 Мова програмування Python

Для розробки системи розпізнавання втомленості водія використовується мова програмування Python. Python – це мова програмування загального призначення, яка має динамічну сувору типізацію, легке керування пам’яттю та спрямована на підвищення продуктивності розробника, а також на розбірливість і якість коду, одночасно дозволяючи переносимість програм.

Унікальна характеристика Python полягає в тому, що все в мові розглядається як об’єкт. Однією з особливостей мови є те, що блоки коду розділені відступами. Синтаксис ядра Python досить стислий, тому він рідко потребує посилання на документацію. Крім того, Python зазвичай використовується для створення сценаріїв через те, що це інтерпретована мова. Однак він також має певні недоліки, такі як повільна продуктивність, а також більш високе споживання пам’яті програмами, написаними з його використанням, порівняно з програмним кодом, написаним на C та C++.

Python – це універсальна мова програмування, яку можна використовувати для широкого спектру цілей завдяки підтримці кількох парадигм програмування. Вона сумісна з парадигмами імперативного, процедурного, структурного, об’єктно-орієнтованого програмування, метапрограмування та функціонального програмування. Python частково підтримує аспектно-орієнтоване програмування за допомогою декораторів, але більш повний спектр підтримки можна знайти за

допомогою додаткових фреймворків. Крім того, Python має бібліотеки та розширення, які дозволяють реалізувати передові методи, такі як контрактне та логічне програмування. Динамічна типізація Python, автоматичне керування пам'яттю, всебічний самоаналіз, механізм обробки винятків, підтримка багатопоточних обчислень із глобальним блокуванням інтерпретатора (GIL) і високорівневі структури даних роблять його високоефективною та зручною мовою програмування. Наразі Python є однією з найпоширеніших мов програмування в багатьох сферах, включаючи аналіз даних, машинне навчання, DevOps, веб-розробку та навіть розробку ігор.

Однією з причин популярності Python є його легкий для читання синтаксис і проста структура, що робить його чудовим вибором для початківців, які навчаються програмувати, або досвідчених програмістів, які хочуть зосередитися на алгоритмах, концепціях і парадигмах. Ще одна перевага Python полягає в тому, що він інтерпретується, що значно полегшує програмістам налагодження та експерименти. Станом на червень 2023 року Python посів перше місце в рейтингу популярності мов програмування [12].

3.1.2 Бібліотека OpenCV

OpenCV – бібліотека програмного забезпечення з відкритим вихідним кодом, розроблена для сфери комп'ютерного зору, обробки зображень і числових алгоритмів. Ця бібліотека містить різноманітний набір інструментів, які можна використовувати для перевірки та маніпулювання вмістом зображень, забезпечуючи оптимізовані та вдосконалені вихідні результати.

У колекції бібліотеки є близько 2500 оптимізованих алгоритмів, включаючи багато різних традиційних і сучасних методів комп'ютерного зору та машинного навчання [9]. Ці алгоритми здатні виявляти та класифікувати обличчя, ідентифікувати об'єкти, класифікувати дії людей у відео, відстежувати рух камери, стежити за рухомими об'єктами, видобувати 3D-моделі об'єктів, отримувати 3D-хмари точок із стереокамер, з'єднувати зображення, знімати

зображення з високою роздільною здатністю всю сцену, пошук пов’язаних зображень у базах даних зображень, видалення червоних очей від спалахів, відстеження очей, розпізнавання сцен тощо.

Численні стартапи, зокрема Applied Minds, VideoSurf і Zeitera, а також такі відомі корпорації, як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda і Toyota, активно використовують OpenCV. Зшивання вуличних зображень, моніторинг шахтного обладнання в Китаї, допомога роботам Willow Garage у навігації та підбиранні об’єктів, апуск інтерактивного мистецтва в Іспанії та Нью-Йорку, перевірки етикеток продуктів на заводах по всьому світу та швидкий догляд за обличчям визнання в Японії – це лише кілька прикладів програм OpenCV.

Бібліотека OpenCV надає інтерфейси для мов програмування C++, Python, Java та MATLAB. Вона сумісний з багатьма операційними системами, включаючи Windows, Linux, Android і Mac OS. OpenCV в основному розроблено для програм комп’ютерного зору в реальному часі та оптимізує продуктивність за допомогою інструкцій MMX і SSE, якщо вони доступні. Крім того, активно розробляються повнофункціональні інтерфейси CUDA і OpenCL. Однією з головних переваг OpenCV є його відкритий вихідний код, який сприяє творчості та співпраці. Він також пропонує високоточні можливості розпізнавання об’єктів і має широкий спектр баз даних, які підтримують функції на основі положення голови. Крім того, OpenCV забезпечує ефективне налагодження, що корисно для розробників, які працюють над проектами комп’ютерного зору. Однак одним із потенційних недоліків OpenCV є середня швидкість обробки даних у сценаріях реального часу. Незважаючи на це, OpenCV залишається широко використовуваною та популярною бібліотекою для розпізнавання об’єктів.

3.1.3 Бібліотека MediaPipe

Google MediaPipe – це програмна основа, випущена в 2019 році, яка дозволяє створювати конвеєрні рішення для ряду даних датчиків. Користувачі можуть створити необхідний вихід, побудувавши граф модульних компонентів

за допомогою конвеєра MediaPipe. Алгоритми обробки та зміни даних для отримання бажаного результату можуть бути включені в ці модульні компоненти. Зручність використання MediaPipe для програм машинного навчання, а також його використання для науковців, студентів і розробників програмного забезпечення є важливим аспектом [22].

Бібліотека MediaPipe розділена на три частини: калькулятор, пакети введення та виведення, а також графіки обчислень. Калькулятор є головним об’єктом, відповідальним за обробку вхідних даних і створення вихідних даних у верхній частині графа обчислень. Кожен вузол у мережі представляє калькулятор, який може приймати та створювати різні вхідні потоки та сторонні пакети. Калькулятор працює, змінюючи вхідні дані за допомогою встановлених функцій, а потім створюючи вихідні дані, які були оброблені належним чином.

Обчислювальний граф складається з вузлів, які працюють як шлюз для проходження пакетів даних. Потік даних усередині графа може надходити від вихідних вузлів, які не отримують жодних вхідних потоків і можуть створювати пакети самостійно. Вхідні потоки також можна використовувати для передачі пакетів даних на граф. Крім того, граф має вузли-приймачі, які приймають пакети даних і зберігають їх у різних місцях. За допомогою Mediapipe можна створити конвеєр, який має такі функції: швидке розпізнавання рухів; ідентифікація 3D об'єктів на фотографіях; знаходити та розпізнавати частини тіла та їх розташування; знаходити та розпізнавати обличчя людей і створювати Face Mesh; знаходити та розпізнавати форми та зіниці окремих людей.

GStreamer служить потужним інструментом, який дозволяє налаштовувати, створюючи спеціальні графіки, які забезпечують ефективну обробку як відео, так і зображень. Хоча GStreamer в основному зосереджений на низькорівневих операціях, призначених для редагування зображень і відео, OpenCV надає альтернативу через створення графіків для визначення послідовності операцій обробки зображень і відео. Однак MediaPipe є більш життєздатним рішенням для аналізу зображень і відео завдяки розширеним можливостям підтримки широкого діапазону типів даних, включаючи дані

часових рядів. Крім того, здатність MediaPipe працювати з семантикою вищого рівня дозволяє створювати кілька виходів завдань з одного входу, що призводить до складних і динамічних відповідей, ефективних у розпізнаванні та аналізі зображень і відеовмісту. Однак такі інструменти, як Beam і Dataflow, зосереджені на обробці великих обсягів даних у великих пакетах на обчислювальних кластерах, що робить їх непридатними для виявлення, розпізнавання та аналізу зображень, відео та даних камери в реальному часі. Коли йдеться про обробку зображень і відео, GStreamer і MediaPipe є двома широко використовуваними інструментами, але вони відрізняються підходами та можливостями.

За допомогою MediaPipe можна легко створювати складні графіки обробки, які складаються з різних завдань, які використовують загальний потік даних. Крім того, MediaPipe підтримує обробку в реальному часі та обробку потокового відео в реальному часі, що робить його ідеальним для розробки рішень, пов'язаних із розпізнаванням образів, доповненою реальністю та відеоаналітикою. Можливості машинного навчання MediaPipe дозволяють створювати моделі, які можуть розпізнавати об'єкти, виконувати семантичну сегментацію та вирішувати інші завдання комп'ютерного зору. Вибір GStreamer, OpenCV або MediaPipe залежить від конкретних потреб, оскільки GStreamer пропонує гнучкість і контроль над операціями низькорівневої обробки зображень і відео, OpenCV дозволяє створювати графіки послідовної обробки, а MediaPipe надає широкий спектр можливостей для зображень і відео.

3.1.4 Бібліотека Tensorflow

TensorFlow – це безкоштовна бібліотека програмного забезпечення для машинного навчання з відкритим кодом. Google розробив його, щоб задовольнити власні потреби у створенні та навчанні нейронних мереж, здатних розшифровувати складні шаблони та зв'язки в даних, подібно до того, як люди навчаються та інтерпретують. TensorFlow використовує графи потоку даних, які складаються з вузлів, що представляють математичні процеси або точки

введення/виведення. Краї цих вузлів вказують на багатовимірні масиви даних (тензори), які можуть перетікати між ними. Ці вузли можуть бути підключені до комп'ютерного обладнання та працювати асинхронно, паралельно обробляючи всі необхідні тензори. Це дозволяє вузлам нейронної мережі бути активними одночасно, аналогічно тому, як нейрони в мозку співпрацюють один з одним.

3.1.5 Бібліотека Keras

Keras – це платформа з відкритим кодом на основі Python, яка спрощує створення нейронних мереж. Його архітектура містить кілька реалізацій стандартних блоків нейронної мережі, таких як цільові та передавальні функції, оптимізатори, згорткові та рекурентні нейронні мережі, а також інструменти для редукції зображень і тексту. Функції Keras допомагають спростити кодування, необхідне для розробки глибокого коду нейронної мережі. Окрім типових нейронних мереж, Keras забезпечує широку підтримку інших основних рівнів обслуговування, таких як нормалізація пакетів, виключення та агрегація. Основні переваги використання Keras включають: простоту використання та швидке розгортання; наявність попередньо навчених моделей (Xception, VGG16, VGG19, ResNet, ResNetV2, MobileNetV2); підтримка графічних процесів.

3.1.6 Бібліотека TorchVision

Бібліотека Torch – це потужний і широко використовуваний інструмент для розробки та навчання глибоких нейронних мереж. Його популярність частково пояснюється простотою використання, ефективністю, гнучкістю та численними функціями. Однією з його найважливіших особливостей є тензорна обчислювальна система, яка дозволяє ефективно маніпулювати багатовимірними масивами даних. Ще однією ключовою особливістю є вбудована підтримка автоматичної диференціації, яка спрощує процес навчання складних моделей глибокого навчання. Модульна структура Torch дозволяє

зручно поєднувати різні компоненти моделі, забезпечуючи гнучкість і можливості налаштування. Крім того, Torch підтримує використання графічних процесорів (GPU) для прискорення обчислень. Однак під час роботи з великими моделями та обсягами даних, особливо з обмеженими ресурсами, динамічна обчислювальна графіка PyTorch може призвести до значного споживання пам'яті. Загалом, бібліотека Torch є чудовим вибором для впровадження високопродуктивних моделей машинного навчання.

3.1.7 Бібліотека Matplotlib

Matplotlib – це бібліотека Python, яка в основному використовується для візуалізації даних за допомогою двовимірної 2D-графіки, а також підтримує 3D-графіку. Це універсальний пакет із можливістю налаштування, який у поєднанні з іншими бібліотеками, такими як NumPy, SciPy та IPython, пропонує можливості, подібні до можливостей MATLAB. Крім того, він сумісний з різними графічними бібліотеками, включаючи wxWindows і PyGTK.

Matplotlib пропонує безліч типів графіків, таких як точкові діаграми, лінійні діаграми, стовпчасті діаграми, гістограми, секторні діаграми, контурні діаграми та градієнтні поля, такі як діаграми сагайдака та спектрограми. Щоб покращити ці графіки, користувачі можуть додавати різні елементи, такі як осі, підписи, легенди, логарифмічні шкали та полярні координати. Крім того, для створення простих тривимірних графіків Matplotlib надає набір інструментів під назвою mplot3d. Matplotlib пропонує користувачам широкий спектр типів графіків і параметрів налаштування для створення інформативних графіків.

3.2 Опис реалізації основних алгоритмів програмної системи

Далі, розглянемо детальніше реалізацію основних алгоритмів програмної системи. На рисунку 3.1 представлена програмна реалізація методу для розрахунку співвідношення сторін очей водія (EAR).

```python
def get_ear(landmarks, refer_idxs, frame_width, frame_height):
    """
    Calculate Eye Aspect Ratio for one eye.

    Args:
        landmarks: (list) Detected landmarks list
        refer_idxs: (list) Index positions of the chosen landmarks
                           in order P1, P2, P3, P4, P5, P6
        frame_width: (int) Width of captured frame
        frame_height: (int) Height of captured frame

    Returns:
        ear: (float) Eye aspect ratio
    """
    try:
        # Compute the euclidean distance between the horizontal
        coords_points = []
        for i in refer_idxs:
            lm = landmarks[i]
            coord = denormalize_coordinates(lm.x, lm.y, frame_width, frame_height)
            coords_points.append(coord)

        # Eye landmark (x, y)-coordinates
        P2_P6 = distance(coords_points[1], coords_points[5])
        P3_P5 = distance(coords_points[2], coords_points[4])
        P1_P4 = distance(coords_points[0], coords_points[3])

        # Compute the eye aspect ratio
        ear = (P2_P6 + P3_P5) / (2.0 * P1_P4)

    except:
        ear = 0.0
        coords_points = None

    return ear, coords_points
```

Рисунок 3.1 – Програмна реалізація алгоритму розрахунку співвідношення сторін очей водія

Метод get_ear() обчислює співвідношення сторін очей ка (EAR) для одного ока на основі виявлених ознак. Метод приймає наступні аргументи:

1. landmarks: Список виявлених ознак обличчя. refer_idxs: Список позицій індексів обраних ознак в порядку P1, P2, P3, P4, P5, P6.

2. frame_width: Ширина захопленого кадру.

3. frame_height: Висота захопленого кадру.

В межах блоку try, метод продовжує обчислювати евклідову відстань між горизонтальними координатами обраних ознак. Ініціалізується список coords_points, який буде зберігати денормалізовані координати ознак. Для

кожного індексу у refer_idxs, метод отримує відповідну ознаку з landmarks, денормалізує її координати за допомогою функції denormalize_coordinates і додає отриману координату до coords_points. Метод обчислює евклідові відстані між певними парами ознак: P2-P6, P3-P5 та P1-P4. Далі, обчислюється співвідношення аспектів ока (ear), поділивши суму відстаней P2-P6 та P3-P5 на двічі відстань P1-P4. Якщо виникають помилки під час виконання (наприклад, помилка при отриманні ознак або обчисленні відстаней), виконується блок except, де змінна ear встановлюється рівною 0.0.

Результатом роботи методу є кортеж, що містить співвідношення аспектів ока (ear) та координати ознак (coords_points).

На рисунку 3.2 представлена реалізація методу, який відповідає за візуалізацію ознак обличчя, що пов’язані з очима водія.

```
def plot_eye_landmarks(frame, left_lm_coordinates, right_lm_coordinates, color):
    for lm_coordinates in [left_lm_coordinates, right_lm_coordinates]:
        if lm_coordinates:
            for coord in lm_coordinates:
                cv2.circle(frame, coord, 2, color, -1)

    frame = cv2.flip(frame, 1)
    return frame
```

Рисунок 3.2 – Програмна реалізація алгоритму побудови ознак очей

Метод приймає наступні аргументи:

frame: Зображення, на якому будуть відображатися ознаки обличчя.

left_lm_coordinates: Координати лівого ока (список координат).

right_lm_coordinates: Координати правого ока (список координат).

color: Колір, яким будуть малюватися ознаки.

У методі здійснюється ітерація по left_lm_coordinates та right_lm_coordinates (ліве і праве око відповідно). Для кожного зі списків координат, перевіряється, чи вони не є пустими (if lm_coordinates). Якщо

координати не є пустими, для кожної координати відбувається наступне: за допомогою cv2.circle малюється коло з центром у вказаній координаті (coord), розмір кола встановлюється на 2 пікселі, а колір кола визначається параметром color. Параметр -1 вказує, що коло повинно бути заповненим.

Після циклу над координатами, зображення frame перевертається горизонтально за допомогою cv2.flip для правильного відображення. Результатом роботи методу є зображення frame, на якому нанесені кола відповідних розмірів та кольорів для лівого та правого ока.

На рисунку 3.3 представлена реалізація основного алгоритму системи для виявлення сонливості водія на основі розрахованих характеристик.

```
if results.multi_face_landmarks:
    landmarks = results.multi_face_landmarks[0].landmark
    EAR, coordinates = calculate_avg_ear(landmarks, self.eye_idxs["left"], self.eye_idxs["right"], frame_w, frame_h)
    frame = plot_eye_landmarks(frame, coordinates[0], coordinates[1], self.state_tracker["COLOR"])

    if EAR < thresholds["EAR_THRESH"]:
        end_time = time.perf_counter()

        self.state_tracker["DROWSY_TIME"] += end_time - self.state_tracker["start_time"]
        self.state_tracker["start_time"] = end_time
        self.state_tracker["COLOR"] = self.RED

        if self.state_tracker["DROWSY_TIME"] >= thresholds["WAIT_TIME"]:
            self.state_tracker["play_alarm"] = True
            plot_text(frame, "ALERT! WAKE UP!", ALM_txt_pos, self.state_tracker["COLOR"])

    else:
        self.state_tracker["start_time"] = time.perf_counter()
        self.state_tracker["DROWSY_TIME"] = 0.0
        self.state_tracker["COLOR"] = self.GREEN
        self.state_tracker["play_alarm"] = False

    EAR_txt = f"EAR: {round(EAR, 2)}"
    DROWSY_TIME_txt = f"DROWSY: {round(self.state_tracker['DROWSY_TIME'], 3)} Secs"
    plot_text(frame, EAR_txt, self.EAR_txt_pos, self.state_tracker["COLOR"])
    plot_text(frame, DROWSY_TIME_txt, DROWSY_TIME_txt_pos, self.state_tracker["COLOR"])

else:
    self.state_tracker["start_time"] = time.perf_counter()
    self.state_tracker["DROWSY_TIME"] = 0.0
    self.state_tracker["COLOR"] = self.GREEN
    self.state_tracker["play_alarm"] = False

    frame = cv2.flip(frame, 1)

return frame, self.state_tracker["play_alarm"]
```

Рисунок 3.3 – Основний програмний алгоритм системи для виявлення сонливості водія на основі розрахованих характеристик

Метод приймає наступні аргументи:frame: Вхідна матриця зображення (np.array). thresholds: Словник, що містить два порогових значення: WAIT_TIME і EAR_THRESH. Спочатку відбувається перевірка на те, чи були знайдені ознаки обличчя (results.multi_face_landmarks не є пустим списком).

Якщо так, відбувається отримання координат обличчя для першого знайденого обличчя (landmarks = results.multi_face_landmarks[0].landmark). Далі, проводиться обчислення середнього значення Eye Aspect Ratio (EAR) для обох очей викликом функції calculate_avg_ear з використанням отриманих координат ознак обличчя та інших параметрів. Викликається функція plot_eye_landmarks для відображення ознак очей на кадрі frame з використанням отриманих координат ознак обличчя та кольору зі стану state_tracker. Проводиться перевірка на те, чи значення Eye Aspect Ratio (EAR) менше порогового значення EAR_THRESH. Якщо так, збільшення значення DROWSY_TIME на різницю між поточним часом і значенням start_time (час початку періоду, коли EAR менше порогового значення), та оновлення start_time для наступної ітерації. Перевірка, чи значення DROWSY_TIME стало більшим або рівним значенню WAIT_TIME. Якщо так, встановлення прапорця play_alarm на True та виклик функції plot_text для відображення тексту "WAKE UP! WAKE UP" на кадрі frame.

Якщо значення Eye Aspect Ratio (EAR) не менше порогового значення EAR_THRESH: Оновлення start_time на поточний час. Скидання значення DROWSY_TIME на 0.0. Встановлення значення play_alarm на False. Якщо ознаки обличчя не були знайдені (results.multi_face_landmarks є пустим списком), то відбувається оновлення start_time на поточний час, скидання значення DROWSY_TIME на 0.0, встановлення прапорця play_alarm на False, а також Виконання горизонтального відображення кадру frame. Метод повертає оброблену матрицю зображення та булеве значення, який показує, чи потрібно відтворювати сигнал тривоги при виявленні сонливості.

Для відтворення сигналу при виявленні сонливості був розроблений окремий клас AudioFrameHandler, який представлений на рисунку 3.4.

```python
def process(self, frame: av.AudioFrame, play_sound: bool = False):

    if not self.audio_segments_created:
        self.prepare_audio(frame)

    raw_samples = frame.to_ndarray()
    _curr_segment = self.play_state_tracker["curr_segment"]

    if play_sound:
        if _curr_segment < self.total_segments:
            _curr_segment += 1
        else:
            _curr_segment = 0

        sound = self.audio_segments[_curr_segment]

    else:
        if -1 < _curr_segment < self.total_segments:
            _curr_segment += 1
            sound = self.audio_segments[_curr_segment]
        else:
            _curr_segment = -1
            sound = AudioSegment(
                data=raw_samples.tobytes(),
                sample_width=frame.format.bytes,
                frame_rate=frame.sample_rate,
                channels=len(frame.layout.channels),
            )
            sound = sound.apply_gain(-100)

    self.play_state_tracker["curr_segment"] = _curr_segment

    channel_sounds = sound.split_to_mono()
    channel_samples = [s.get_array_of_samples() for s in channel_sounds]

    new_samples = np.array(channel_samples).T

    new_samples = new_samples.reshape(self.audio_segment_shape)
    new_frame = av.AudioFrame.from_ndarray(new_samples, layout=frame.layout.name)
    new_frame.sample_rate = frame.sample_rate

    return new_frame
```

Рисунок 3.4 – Програмна реалізація алгоритму обробки аудіопотоків

Метод process приймає поточний аудіофрейм frame та на підставі значення булевої змінної play_sound або починає відтворювати звуковий сегмент, або знижує амплітуду звукового сигналу, щоб імітувати тишу.

Спочатку відбувається перевірка, чи були створені аудіосегменти (audio_segments_created є істинним значенням). Якщо ні, виклик методу

prepare_audio для підготовки аудіосегментів на основі вхідного аудіофрейму frame. Конвертація аудіофрейму frame в масив сирих зразків за допомогою frame.to_ndarray(). Отримання поточного сегмента зі стану play_state_tracker (_curr_segment). Далі, якщо play_sound є істинним значенням: відбувається перевірка, чи поточний сегмент _curr_segment менше за загальну кількість сегментів total_segments. Якщо так, збільшення значення _curr_segment на 1. В іншому випадку, присвоєння _curr_segment значення 0. Відбувається отримання звукового сегменту sound зі списку аудіосегментів audio_segments.

В іншому випадку (якщо play_sound є хибним значенням) відбувається перевірка, чи значення _curr_segment знаходиться в діапазоні від -1 до total_segments. Якщо так, збільшення значення _curr_segment на 1. В іншому випадку, присвоєння значення _curr_segment -1 (позначає, що не відтворюється жоден сегмент). Відбувається створення сегменту sound на основі сирих зразків з аудіофрейму frame, знижуючи гучність на -100 децибел.

Проводиться розбиття звукового сегменту sound на окремі монофоні канали за допомогою sound.split_to_mono(). Відбувається отримання масивів зразків з кожного каналу звукового сегменту. Перетворення масивів зразків каналів в двовимірний масив new_samples, де кожен рядок представляє зразки з відповідного каналу. Зміна форми new_samples на audio_segment_shape. Створення нового аудіофрейму new_frame з масиву new_samples з використанням av.AudioFrame.from_ndarray(). Встановлення значення частоти дискретизації sample_rate фрейму на значення частоти дискретизації frame. В результаті відбувається повернення нового аудіофрейму new_frame.

Висновки до розділу 3

В третьому розділі представлено опис мови програмування Python, систематизовані переваги і недоліки її використання для розробки програмної системи виявлення рівня втомленості людини. Обгрунтовано використання бібліотек Tensorflow, Keras, PyTorch для розробки моделі нейронної мережі.

4. ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ПРОГРАМНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ВТОМЛЕНОСТІ ВОДІЯ

4.1 Системні вимоги

Розроблений застосунок для розпізнавання втомленості водія призначений для використання на пристроях з ОС Windows, MacOS, Linux. Користувач повинен мати високу роздільна здатність камери або іншого пристрою для захоплення зображення обличчя водія. Також, важливим елементом є достатня потужність графічної підсистеми для плавного відображення та візуалізації результатів розпізнавання втомленості. Для запуску застосунку, система повинна мати не менше 2 ГБ вільної системної пам’яті для встановлення залежностей та не менше 16 ГБ оперативної пам’яті.

4.2 Запуск системи

Для запуску системи користувачу необхідно:

– Встановити інтерпретатор Python версії вище 3.9.0.

– Розархівувати проєкт та перейти у корневий каталог.

– Створити віртуальне середовище для роботи (python -m venv venv).

– Активувати віртуальне середовище (source venv/bin/activate).

– Встановити залежності для роботи системи (pip install -r requirements.txt).

– Запустити застосунок (streamlit run app.py –port 3000).

4.3 Тестування програмної системи

На рисунку 4.1 представлений головний вигляд системи розпізнавання втомленості водія. Для початку роботи користувачу необхідно ввімкнути камеру для активації системи детектування сонливості.

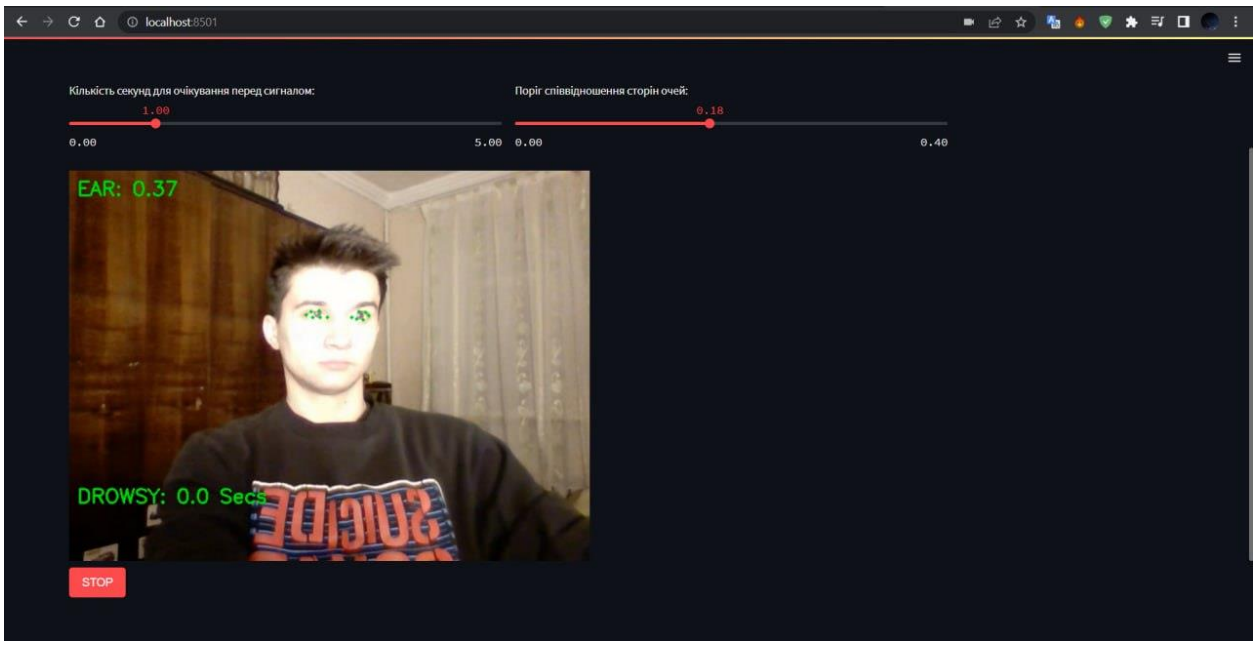


Рисунок 4.1 – Вигляд головної сторінки системи розпізнавання

На рисунку 4.2 представлений результат розпізнавання та детектування втомленості людини та відповідне сповіщення.

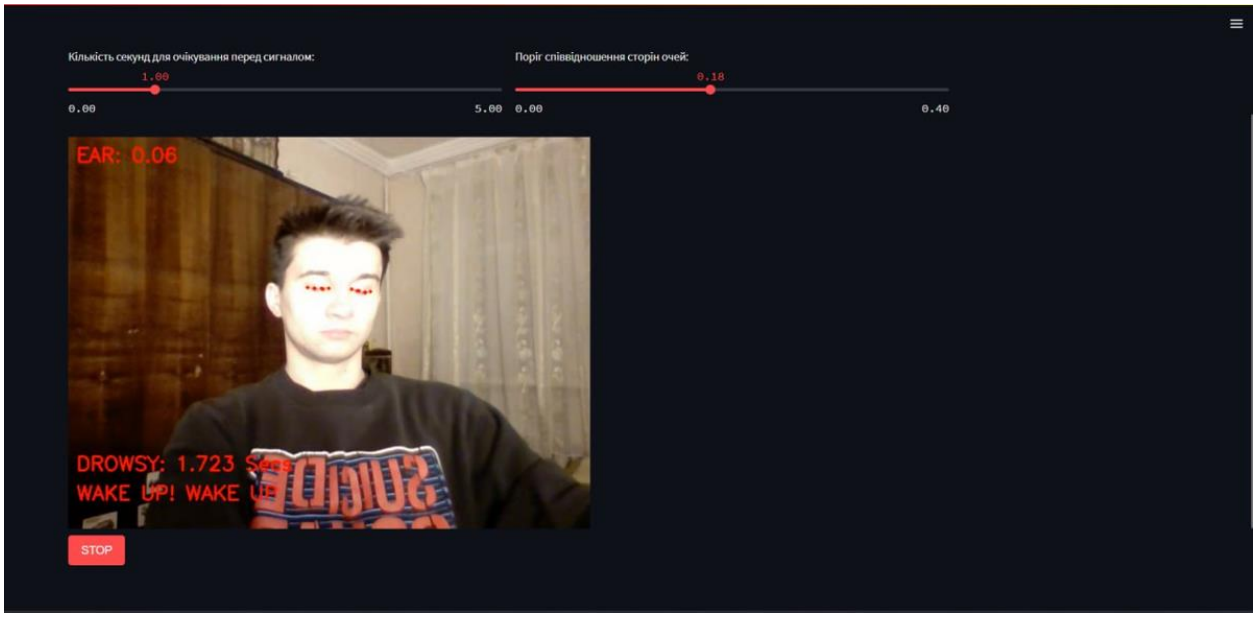


Рисунок 4.2 – Результат розпізнавання сонливості людини та попередження про перевищення порогового значення

Для налаштування доступні два параметри:

1. Кількість секунд очікування перед активацією аудіосигналу попередження водія. За замовчуванням має значення – 1 секунда.

2. Поріг співвідношення сторін очей – визначає порогове значення для співвідношення сторін очей (EAR). При досягненні або перевищенні цього порогового значення система може ввімкнути сигнал попередження водія.

Після того як водій відкриває очі, система знову перевіряє риси обличчя. Програмні константи та змінні приймають значення за замовчуванням. У випадку, якщо очі людини залишаються закритими більше ніж WAIT_TIME кадрів, активується звуковий сигнал. Завдяки експериментам значення THRESH_EAR 0,2 дало сприятливі результати в різних налаштуваннях. Для запуску аудіосигналу значення WAIT_TIME було встановлено експериментально на 1 секунду (приблизно 48 кадрів) – якщо очі людини закриті протягом 48 послідовних кадрів, буде спрацьовувати звуковий сигнал. Чутливість детектора можна регулювати, зменшуючи WAIT_TIME для підвищення чутливості або збільшуючи його для меншої чутливості. Обчислення евклідових відстаней відбувається негайно та не залежить від хмарних баз даних, що дає перевагу над існуючими системами. Крім того, швидкість роботи системи відповідає поставленим перед нею завданням.

Поточна система становить потенційну небезпеку, коли надмірно виснажений водій може не реагувати на попереджувальний звуковий сигнал від системи. Щоб вирішити цю проблему, було запропоновано розширити систему, включивши механізм вібрації на сидінні водія, який би сповіщав водія про закриті очі. Крім того, система також зменшувала б оберти двигуна, щоб додатково спонукати водія вжити відповідних дій. Вищезазначені засоби забезпечення безпеки водія є опціональними для впровадження.

Висновки до розділу 4

В четвертому розділі проведено тестування розробленої системи виявлення втомленості водія. На цьому етапі розробки оцінка запропонованої системи підтвердила, що вона відповідає встановленим вимогам. Ризики, пов’язані з системою, були виявлені та усунені за допомогою рішень, що забезпечують безперебійну роботу. Система пройшла ретельне тестування, і всі окремі функції перевірені на функціональність. Під час тестування не було виявлено жодних помилок чи несправностей. Крім того, тести швидкості системи, які проводилися шляхом вимірювання часу від запуску системи до кінцевої стадії, показали очікувані результати. На основі вищенаведеного визначено, що запропонована система підвищує продуктивність аналогів. Щоб покращити продуктивність алгоритму, система постійно відстежує рухи очей без постійного зв’язку з базами даних, що підвищує швидкість роботи.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи розроблено систему виявлення рівня втомленості водія, яка може бути особливо корисною для довгих відрядних поїздок, де ризик сонливості для водіїв збільшується.

У першому розділі проведено аналіз предметної області, визначено основні аналогічні системи, систематизовані переваги і недоліки кожної з них. Проведено постановку задачі на розробку програмної системи розпізнавання сонливості водія, визначено функціональні та нефункціональні вимоги.

В другому розділі проведено дослідження основних алгоритмів, які використовуються при розробці системи розпізнавання втомленості водія. Визначено, що метод головних компонентів використовується для перетворення набору зображень індивідів у загальну матрицю даних у декілька етапів. Цей процес передбачає розміщення зображення кожної людини в рядку в матриці після зміни розміру до однакового розміру та нормалізації їхніх гістограм. Однак ефективність методу головного компонента може бути обмежена в сценаріях, коли на зображенні існують значні варіації в освітленні або виразі обличчя. Це пояснюється тим, що методика покладається на апроксимацію вхідного набору даних, а не на розрізнення різних класів осіб. Це означає, що метод може бути нездатним точно диференціювати людей за наявності великих варіацій у вхідних даних. Проаналізовані дескриптори та детектори SURF, BRISK, SIFT, FAST.

В третьому розділі представлено опис мови програмування Python, систематизовані переваги і недоліки її використання для розробки програмної системи виявлення рівня втомленості людини. Обгрунтовано використання бібліотек Tensorflow, Keras, PyTorch для розробки моделі нейронної мережі.

В четвертому розділі проведено тестування розробленої системи виявлення втомленості водія. На цьому етапі розробки оцінка запропонованої системи підтвердила, що вона відповідає встановленим вимогам. Ризики, пов’язані з системою, були виявлені та усунені за допомогою рішень, що забезпечують безперебійну роботу. Система пройшла ретельне тестування, і всі

окремі функції перевірені на функціональність. Під час тестування не було виявлено жодних помилок чи несправностей. Крім того, тести швидкості системи, які проводилися шляхом вимірювання часу від запуску системи до кінцевої стадії, показали очікувані результати. На основі вищенаведеного визначено, що запропонована система підвищує продуктивність аналогів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tianyi Hong and Huabiao Qin. Drivers drowsiness detection in embedded system. International Conference on Vehicular Electronics and Safety. 2007. pp. 1-5. URL: https://ieeexplore.ieee.org/document/44563 (дата звернення: 26.05.2023).

2. M. S. Basit, U. Ahmad, J. Ahmad, K. Ijaz and S. F. Ali. Driver Drowsiness Detection with Region-of-Interest Selection Based on Deep Convolutional-LSTM. International Conference on Technologies. 2022. pp. 1-6. URL: https://ieeexplore.ieee.org/document/10016825 (дата звернення: 22.05.2023).

3. M. Siwach, S. Mann and D. Gupta. A Practical Implementation of Driver Drowsiness Detection Using Facial Landmarks. International Conference on Reliability, Infocom Technologies and Optimization. 2022. pp. 1-4. URL: https://ieeexplore.ieee.org/document/9964990 (дата звернення: 28.05.2023).

4. D. -L. Nguyen, M. D. Putro. Eyes Status Detector Based on Light-weight Convolutional Neural Networks supporting for Drowsiness Detection System. The 46th Annual Conference of the IEEE Industrial Electronics. 2020. pp. 477-482. URL: https://ieeexplore.ieee.org/document/9254858 (дата звернення: 28.05.2023).

5. S. S, N. Banupriya, S. M and S. N. H. Drowsiness Detection with OpenCV. International Conference on Communication Systems. 2021. pp. 1421-1425. URL: https://ieeexplore.ieee.org/document/9532758 (дата звернення: 28.05.2023).

6. Bajaj, J.S, Natarajan, R. System and Method for Driver Drowsiness Detection Using Behavioral and Sensor-Based Physiological Measures. 2022. URL: https://www.mdpi.com/1424-8220/23/3/1292 (дата звернення: 24.05.2023).

7. Driver Drowsiness Detection. URL: https://www.bosch-mobility.com/solutions/driver-drowsiness-detection (дата звернення: 26.05.2023).

8. J. Flores-Monroy, M. Nakano-Miyatake, G. Sanchez-Perez and H. Perez-Meana. Visual-based Real Time Driver Drowsiness Detection System Using CNN. International Conference on Electrical Engineering. 2021. pp. 1-5. URL: https://ieeexplore.ieee.org/document/9633082 (дата звернення: 28.05.2023).

9. M. S. Devi, M. V. Choudhari and P. Bajaj. Driver Drowsiness Detection Using Skin Color Algorithm and Circular Hough Transform. International Conference on Emerging Trends in Engineering & Technology. 2011. pp. 129-134. URL: https://ieeexplore.ieee.org/document/6120568 (дата звернення: 03.05.2023).

10. Albadawi, Y., Takruri, M., & Awad, M. A Review of Recent Developments in Driver Drowsiness Detection Systems. Sensors. 2022. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/8914892 (дата звернення: 06.05.2023).

11. M Arunasalam, N. Yaakob, A. Amir, M. Elshaikh and N. F. Azahar. Real-Time Drowsiness Detection System for Driver Monitoring. URL: https://iopscience.iop.org/article/1757/767/1/012066 (дата звернення: 08.05.2023).

12. P. Singh, S. Mahim and S. Prakash. Drowsiness Detection for Driver Assistant System. International Conference on Signal Processing. 2022. pp. 1-4. URL: https://ieeexplore.ieee.org/document/1000754 (дата звернення: 08.05.2023).

13. P. Singh, S. P. S. Chauhan and E. Rajesh. Real-Time Driver Drowsiness Detection Using Dlib And openCV. International Conference on Advances in Computing, Communication Control and Networking. 2022. pp. 956-960. URL: https://ieeexplore.ieee.org/document/10074245 (дата звернення: 06.05.2023).

14. A. Al Redhaei, Y. Albadawi, S. Mohamed and A. Alnoman. Realtime Driver Drowsiness Detection Using Machine Learning. Advances in Science and Engineering Technology International Conferences. 2022. pp. 1-6. URL: https://ieeexplore.ieee.org/document/9734801 (дата звернення: 06.05.2023).

15. M. Babaeian, N. Bhardwaj, B. Esquivel and M. Mozumdar. Real time driver drowsiness detection using a logistic-regression-based machine learning algorithm. Green Energy and Systems Conference (IGSEC). 2016. pp. 1-6. URL: https://ieeexplore.ieee.org/document/7790075 (дата звернення: 09.06.2023).

16. I. Latreche, S. Slatnia and O. Kazar. CNN-LSTM To Identify The Most Informative EEG-Based Driver Drowsiness Detection Brain Region. International Symposium on Multidisciplinary Innovative Technologies. 2022. pp. 725-730. URL: https://ieeexplore.ieee.org/document/9932696 (дата звернення: 10.06.2023).

ДОДАТОК 1

До дипломного проєкту

на тему: «Програмна система виявлення сонливості водія на основі OpenCV»

Копії графічних Матеріалів

Діаграма варіантів використання програмної.

ІАЛЦ.045490.005 Д1.

Діаграма послідовності програмного забезпечення.

ІАЛЦ.045490.006 Д2.

Схема роботи алгоритму виявлення втомленості водія.

ІАЛЦ.045490.007 Д3.

Структурна схема алгоритму головних компонент.

ІАЛЦ.045490.008 Д4.

Аркушів 4

Київ – 2023

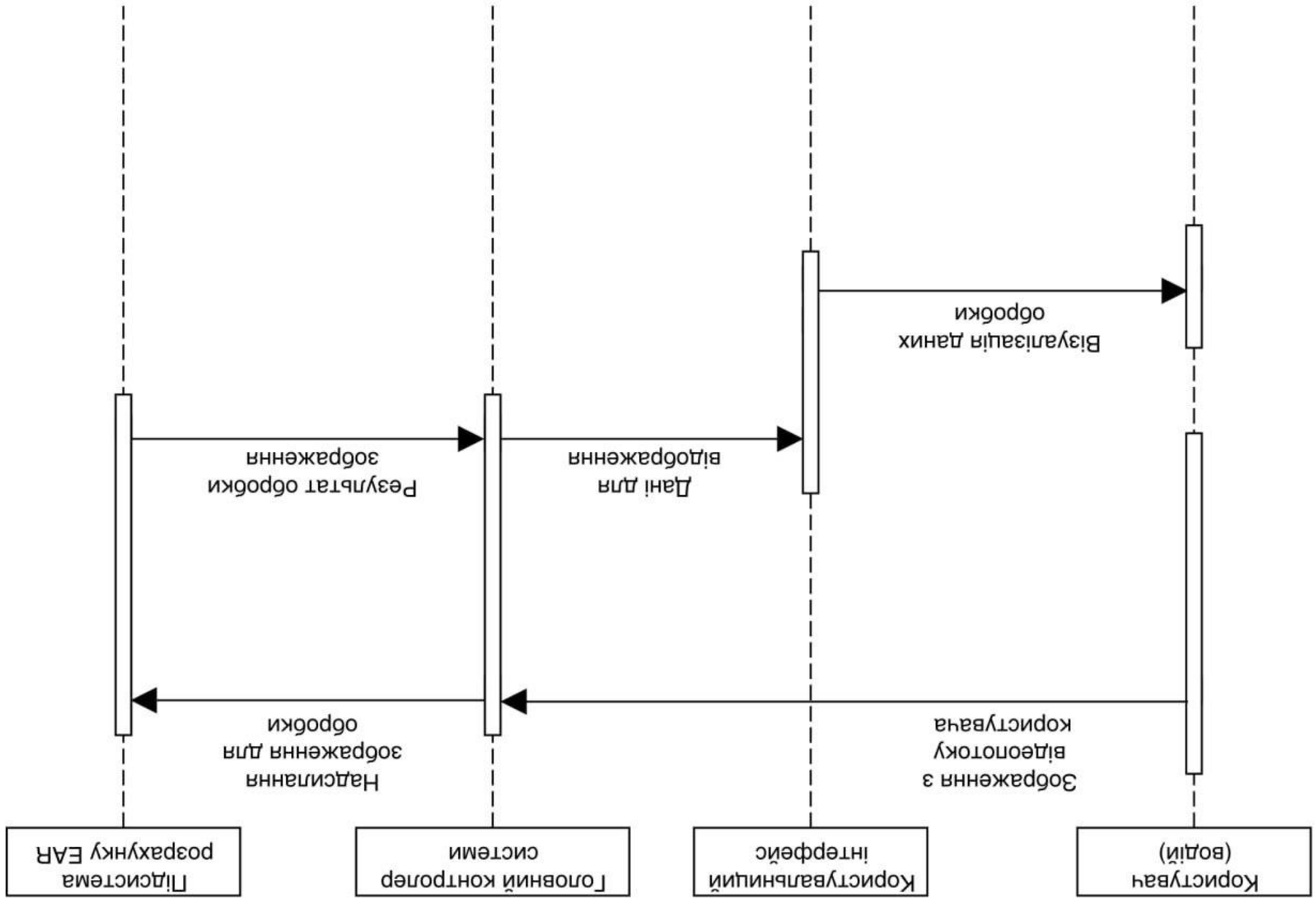
Зм.	Арк.	№ докум.	Підпис	Дата
Розробив		Гула М. Р.		
Перевірив		Петрашенко А. В.		

ІАЛЦ.045490.005 Д1

Програмна система виявлення сонливості водія на основі OpenCV.
Діаграма варіантів використання програмної системи

Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського
Група КВ-92

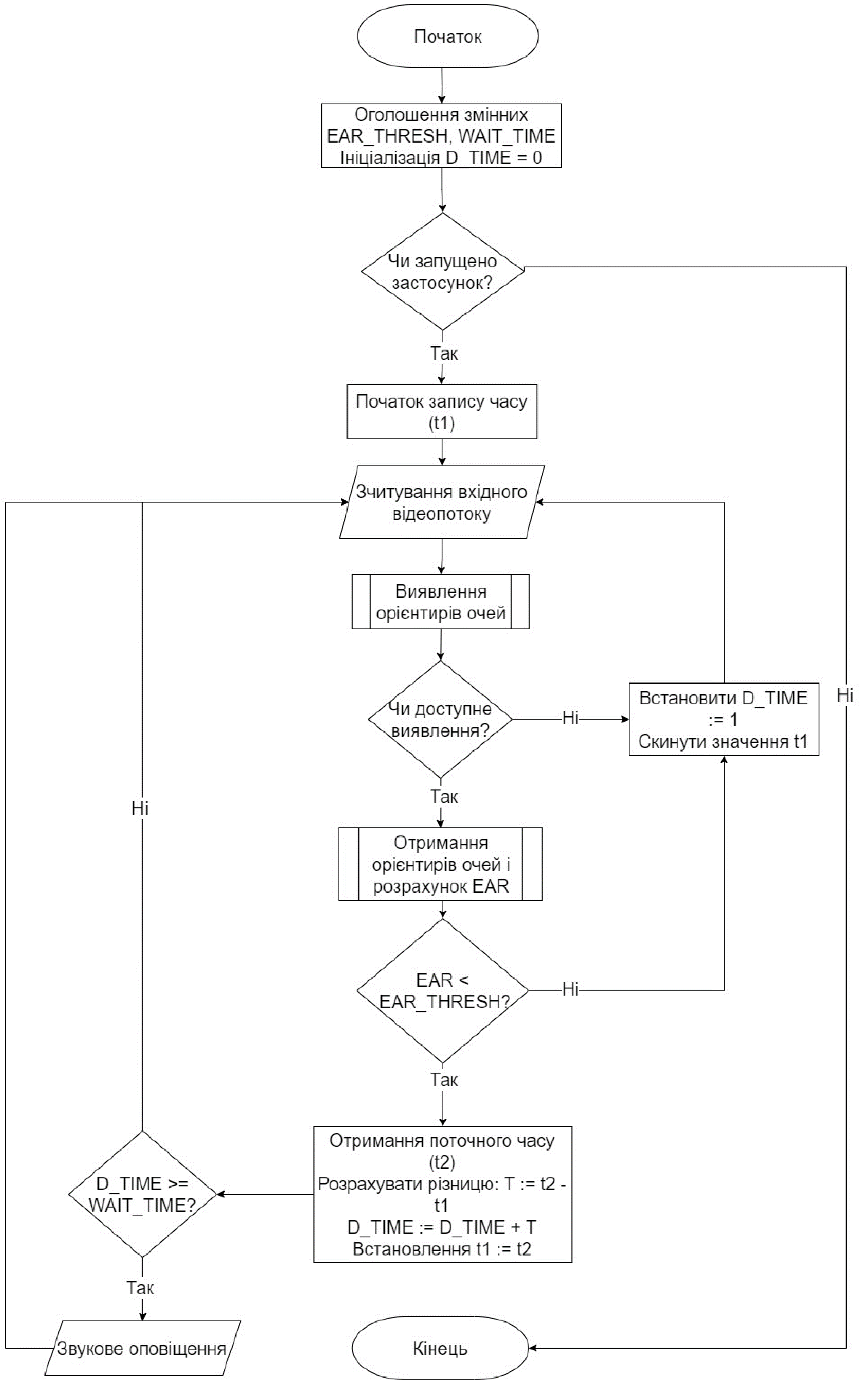


ІАЛЦ.045490.00

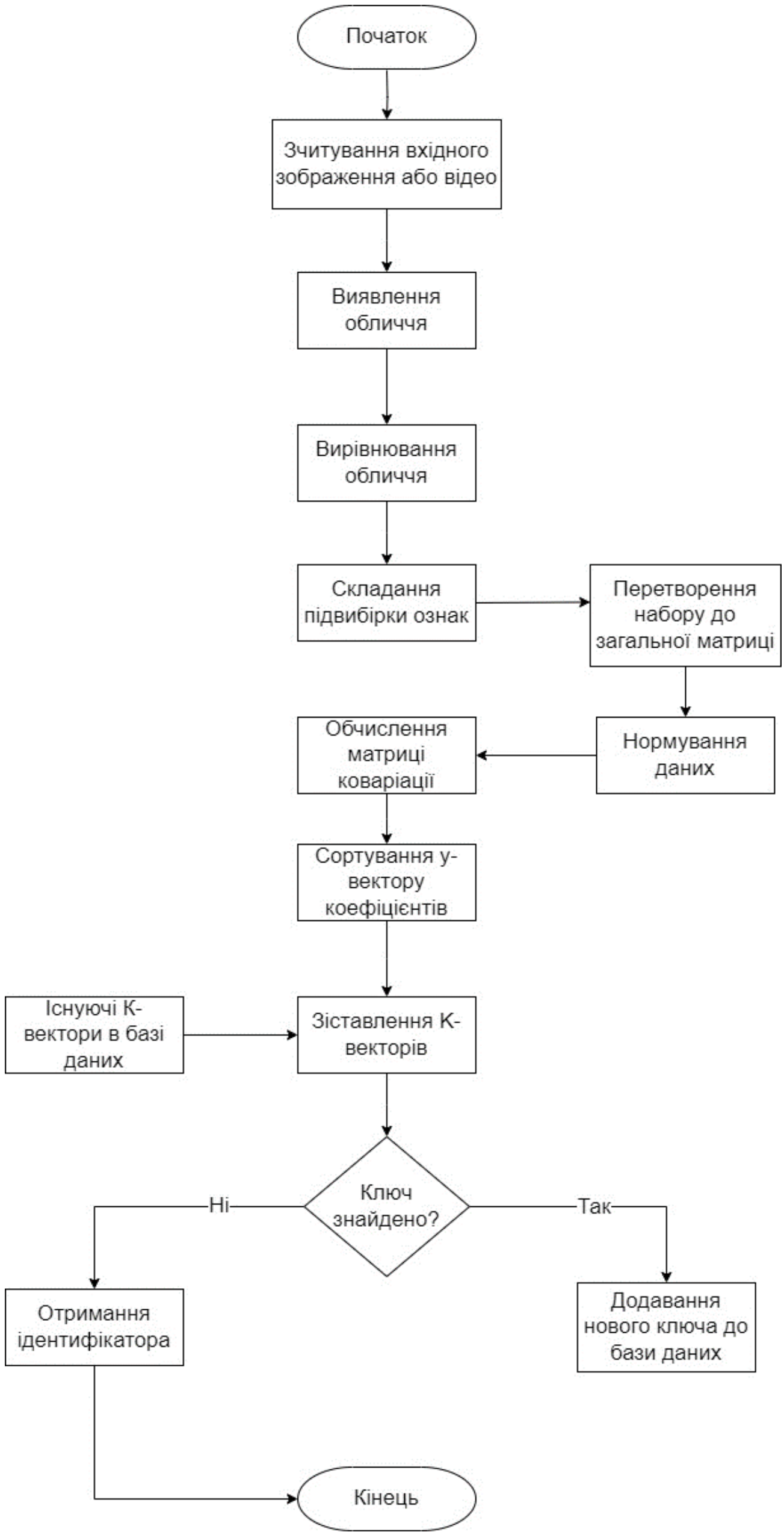
Зм.	Арк.	№ докум.	Підпис	Дата	Програмна система виявлення спільності водіїв на основі OpenCV. *Діаграма послідовності прецедентів завдання*			
Розробив		Гуна М. Р.				Літера	Маса	Масштаб
Перевірив		Петрашенко А. В.						
						Аркуш 1	Аркушів 1	
						КПІ ім. Ігоря Сікорського Група КВ-92		

ІАЛЦ.045490.006 Д2

ІАЛЦ.045490.00



Зм.	Арк.	№ Докум.	Підпис	Дата	ІАЛЦ.045490.007 ДЗ
Розробив		Гула М. Р.			Програмна система виявлення сонливості водія на основі OpenCV. *Схема роботи алгоритму виявлення втомленості водія*
Перевірив.		Петрашенко А. В.			Літера / Маса / Масштаб
					Аркуш 1 / Аркушів 1
					КПІ ім. Ігоря Сікорського Група КВ-92



Зм.	Арк.	№ Докум.	Підпис	Дата
Розробив		Гула М. Р.		
Перевірив.		Петращенко А. В.		

ІАЛЦ.045490.008 Д4

Програмна система виявлення сонливості водія на основі OpenCV.
Структурна схема алгоритму головних компонент

Літера	Маса	Масштаб

Аркуш 1	Аркушів 1

КПІ ім. Ігоря Сікорського
Група КВ-92

ДОДАТОК 2

До дипломного проєкту

на тему: «Програмна система виявлення сонливості водія на основі OpenCV»

Презентація

ІАЛЦ.045490.009 Д5.

Аркушів

Київ – 2023

ПРОГРАМНА СИСТЕМА ВИЯВЛЕННЯ СОНЛИВОСТІ ВОДІЯ НА ОСНОВІ OPENCV

Презентація до кваліфікаційної роботи

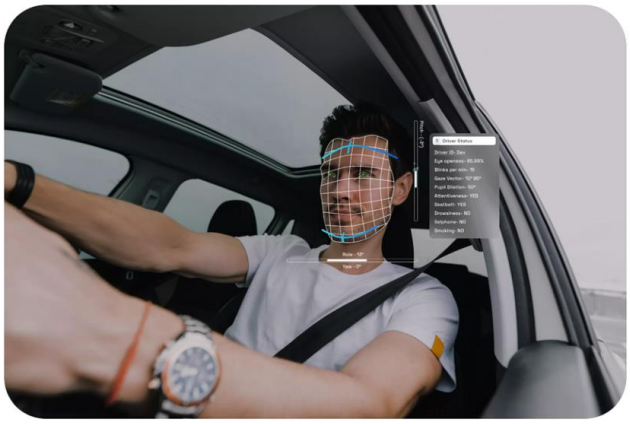
Підготував: Гула Михайло, гр. КВ-92
Керівник: к.т.н, доц.каф.СПСКС
Петрашенко А. В.

ВСТУП

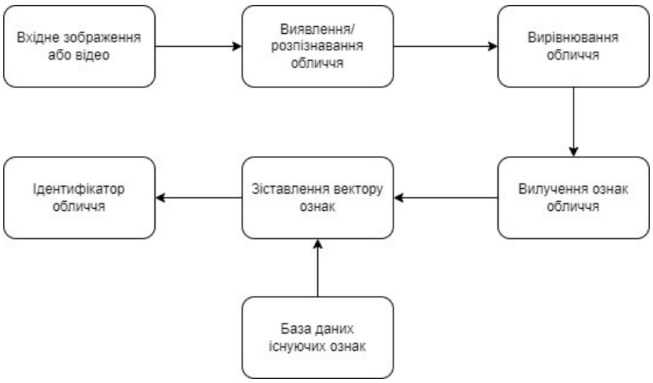
Удосконалення технологій, спрямованих на виявлення та запобігання сонливості під час керування автомобілем, створює серйозну проблему для вдосконалення систем запобігання аварій. Вкрай важливо попередити водія в момент виявлення сонливості, щоб запобігти будь-яким потенційним нещасним випадкам. Стан очей водія може бути засобом виявлення сонливості, тому необхідно уважно стежити за будь-якими ознаками втоми. Ця технологія може відігравати життєво важливу роль у забезпеченні безпеки водія на дорозі, особливо під час тривалих поїздок.

СИСТЕМА ВИЗНАЧЕННЯ ВТОМИ ВОДІЯ

Система контролю втоми відповідає за моніторинг фізичного стану водія з метою забезпечення його безпеки. Вона виявляє будь-які відхилення від очікуваного стандарту та попереджає водія про необхідність зупинитися та зробити перерву, якщо це необхідно. Залежно від методу оцінки втоми водія доступні три різні типи систем. Перший спосіб контролює дії водія, а другий – рух автомобіля.
Нарешті, третій метод покладається на спостереження за поглядом водія, щоб виявити будь-які ознаки сонливості чи втоми.



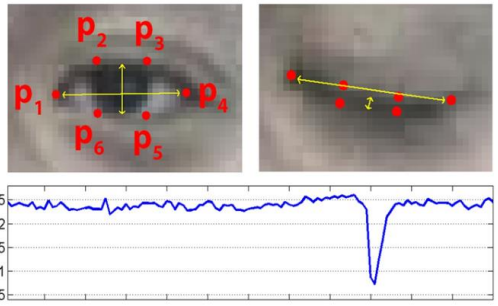
ПРОЦЕС РОЗПІЗНАВАННЯ ОБЛИЧЧЯ ВОДІЯ



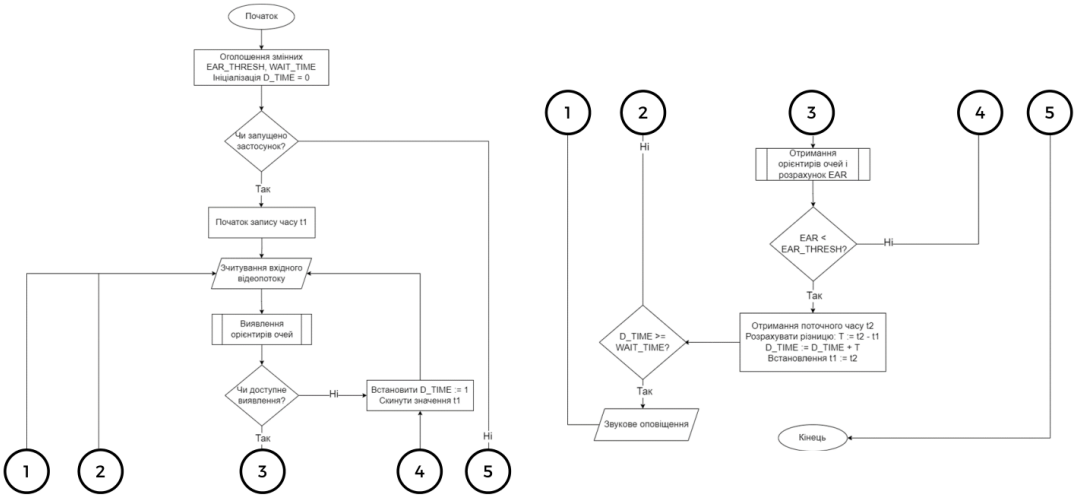
ПРОЦЕС ВИЗНАЧЕННЯ СОНЛИВОСТІ ВОДІЯ

Для визначення стану очей буде використана математична формула, відома як співвідношення сторін очей (**EAR**).
Формула **EAR** повертає скалярну величину, яка відображає рівень відкриття очей. На рисунку представлена візуалізація орієнтирів очей, а також графік співвідношення сторін очей людини з часом.

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

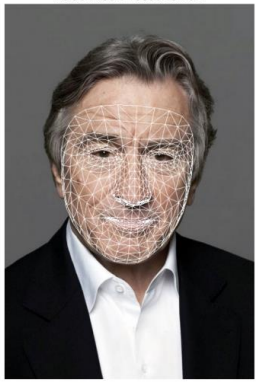


АЛГОРИТМ РОБОТИ СИСТЕМИ

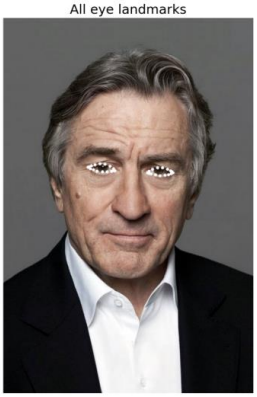


РОЗПІЗНАВАННЯ КЛЮЧОВИХ ТОЧОК ОБЛИЧЧЯ З ВИКОРИСТАННЯМ MEDIAPIPE

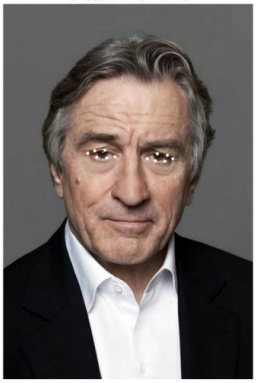
Face Mesh Tessallation



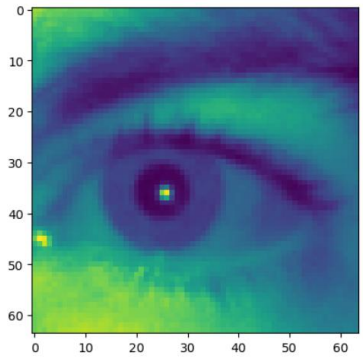
All eye landmarks



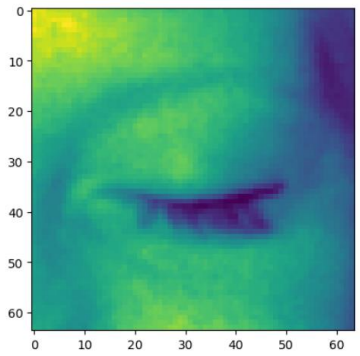
Chosen landmarks



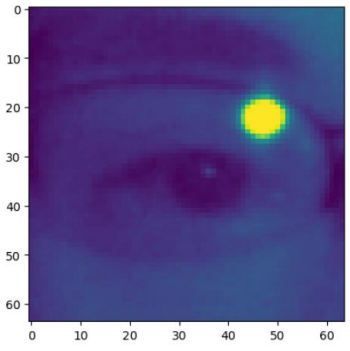
РЕЗУЛЬТАТИ РОЗПІЗНАВАННЯ ОЧЕЙ ВОДІЯ



Open
1/1 [==============================] - 0s 34ms/step

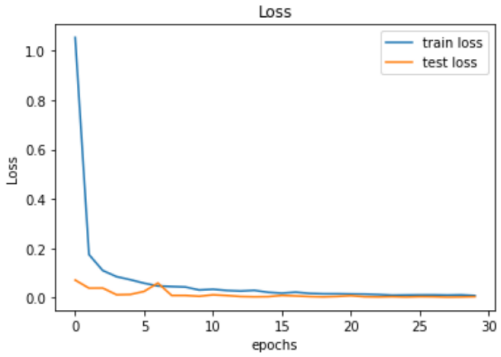


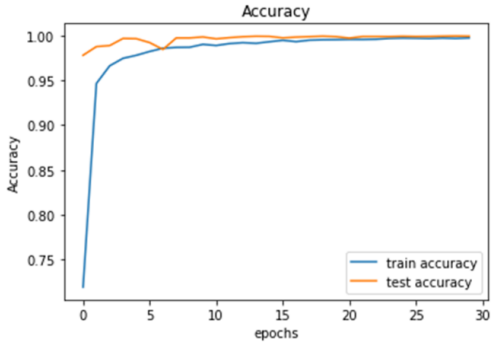
Closed
1/1 [==============================] - 0s 31ms/step



Open
1/1 [==============================] - 0s 34ms/step

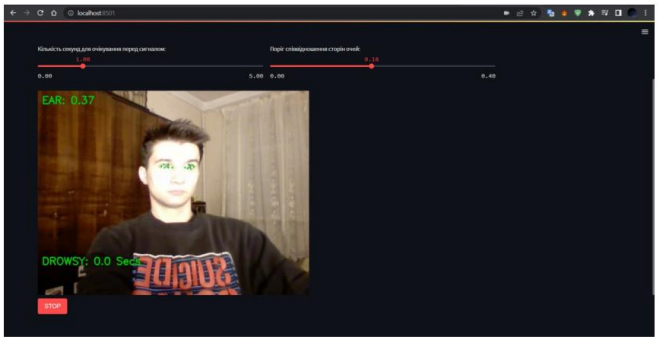
МЕТРИКИ ТОЧНОСТІ ТА ВТРАТ НЕЙРОННОЇ МЕРЕЖІ





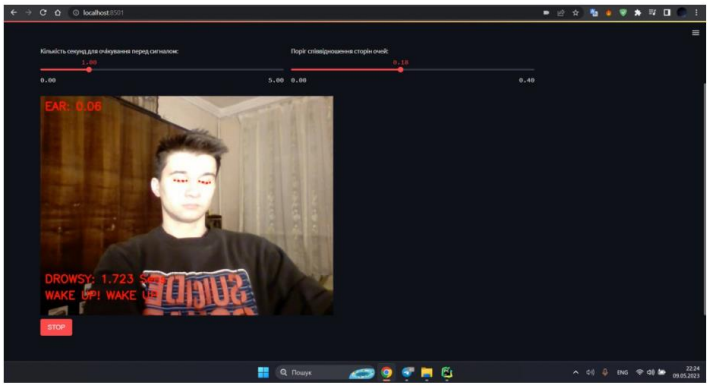
ВИГЛЯД ІНТЕРФЕЙСУ СИСТЕМИ

На рисунку представлено початковий вигляд інтерфейсу системи, коли стан сонливості не ідентифіковано.
Система підтримує налаштування параметру кількості секунд очікування перед подання аудіосигналу для пробудження.



ВИГЛЯД ІНТЕРФЕЙСУ СИСТЕМИ

Як видно з рисунку, знизу зліва система інформує водія про стан сонливості та запускає аудіосигнал для пробудження.



ВИСНОВКИ

Розробка програмної системи, здатної виявляти сонливість водіїв, має велике значення для підвищення безпеки на дорозі. Використання технологій комп'ютерного зору та нейронних мереж на основі **OpenCV** дозволяє точно визначати ознаки сонливості, такі як позіхання та зниження руху очей. Така система може бути використана в різних типах транспортних засобів, включаючи автомобілі, автобуси та вантажівки. Вона може бути особливо корисною для довгих відрядних поїздок, де ризик сонливості збільшується. В результаті проведеного дослідження розроблено нейронну мережу з подальшою інтеграцією в веб-застосунок для спрощення взаємодії користувача з системою. Виявлено, що розроблена модель досягає точності **98%** на тестових даних.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК З

До дипломного проєкту

на тему: «Програмна система виявлення сонливості водія на основі OpenCV»

Лістинг програми

ІАЛЦ.045490.010 Д6.

Аркушів

Київ – 2023

```
import cv2
import time
import numpy as np
import mediapipe as mp
from mediapipe.python.solutions.drawing_utils import
_normalized_to_pixel_coordinates as denormalize_coordinates


def get_mediapipe_app(
    max_num_faces=1,
    refine_landmarks=True,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5,
):
    face_mesh = mp.solutions.face_mesh.FaceMesh(
        max_num_faces=max_num_faces,
        refine_landmarks=refine_landmarks,
        min_detection_confidence=min_detection_confidence,
        min_tracking_confidence=min_tracking_confidence,
    )

    return face_mesh


def distance(point_1, point_2):
    """Calculate l2-norm between two points"""
    dist = sum([(i - j) ** 2 for i, j in zip(point_1, point_2)]) ** 0.5
    return dist
```

```
def get_ear(landmarks, refer_idxs, frame_width, frame_height):
    try:
        coords_points = []
        for i in refer_idxs:
            lm = landmarks[i]
            coord = denormalize_coordinates(lm.x, lm.y, frame_width, frame_height)
            coords_points.append(coord)

        # Eye landmark (x, y)-coordinates
        P2_P6 = distance(coords_points[1], coords_points[5])
        P3_P5 = distance(coords_points[2], coords_points[4])
        P1_P4 = distance(coords_points[0], coords_points[3])

        # Compute the eye aspect ratio
        ear = (P2_P6 + P3_P5) / (2.0 * P1_P4)

    except:
        ear = 0.0
        coords_points = None

    return ear, coords_points


def calculate_avg_ear(landmarks, left_eye_idxs, right_eye_idxs, image_w,
image_h):
    # Calculate Eye aspect ratio

    left_ear, left_lm_coordinates = get_ear(landmarks, left_eye_idxs, image_w,
image_h)
```

```
    right_ear, right_lm_coordinates = get_ear(landmarks, right_eye_idxs, image_w,
image_h)
    Avg_EAR = (left_ear + right_ear) / 2.0

    return Avg_EAR, (left_lm_coordinates, right_lm_coordinates)


def plot_eye_landmarks(frame, left_lm_coordinates, right_lm_coordinates, color):
    for lm_coordinates in [left_lm_coordinates, right_lm_coordinates]:
        if lm_coordinates:
            for coord in lm_coordinates:
                cv2.circle(frame, coord, 2, color, -1)

    frame = cv2.flip(frame, 1)
    return frame


def plot_text(image, text, origin, color, font=cv2.FONT_HERSHEY_SIMPLEX,
fntScale=0.8, thickness=2):
    image = cv2.putText(image, text, origin, font, fntScale, color, thickness)
    return image


class VideoFrameHandler:
    def __init__(self):
        # Left and right eye chosen landmarks.
        self.eye_idxs = {
            "left": [362, 385, 387, 263, 373, 380],
            "right": [33, 160, 158, 133, 153, 144],
        }
```

```
        # Used for coloring landmark points.
        # Its value depends on the current EAR value.
        self.RED = (0, 0, 255)  # BGR
        self.GREEN = (0, 255, 0)  # BGR

        # Initializing Mediapipe FaceMesh solution pipeline
        self.facemesh_model = get_mediapipe_app()

        # For tracking counters and sharing states in and out of callbacks.
        self.state_tracker = {
            "start_time": time.perf_counter(),
            "DROWSY_TIME": 0.0,  # Holds the amount of time passed with EAR <
EAR_THRESH
            "COLOR": self.GREEN,
            "play_alarm": False,
        }

        self.EAR_txt_pos = (10, 30)

    def process(self, frame: np.array, thresholds: dict):

        # To improve performance,
        # mark the frame as not writeable to pass by reference.
        frame.flags.writeable = False
        frame_h, frame_w, _ = frame.shape

        DROWSY_TIME_txt_pos = (10, int(frame_h // 2 * 1.7))
        ALM_txt_pos = (10, int(frame_h // 2 * 1.85))
```

```
        results = self.facemesh_model.process(frame)

        if results.multi_face_landmarks:
            landmarks = results.multi_face_landmarks[0].landmark
            EAR, coordinates = calculate_avg_ear(landmarks, self.eye_idxs["left"],
self.eye_idxs["right"], frame_w, frame_h)
            frame = plot_eye_landmarks(frame, coordinates[0], coordinates[1],
self.state_tracker["COLOR"])

            if EAR < thresholds["EAR_THRESH"]:

                # Increase DROWSY_TIME to track the time period with EAR less than
the threshold
                # and reset the start_time for the next iteration.
                end_time = time.perf_counter()

                self.state_tracker["DROWSY_TIME"] += end_time -
self.state_tracker["start_time"]
                self.state_tracker["start_time"] = end_time
                self.state_tracker["COLOR"] = self.RED

                if self.state_tracker["DROWSY_TIME"] >=
thresholds["WAIT_TIME"]:
                    self.state_tracker["play_alarm"] = True
                    plot_text(frame, "ALERT! WAKE UP!", ALM_txt_pos,
self.state_tracker["COLOR"])

            else:
                self.state_tracker["start_time"] = time.perf_counter()
                self.state_tracker["DROWSY_TIME"] = 0.0
```

```
                self.state_tracker["COLOR"] = self.GREEN
                self.state_tracker["play_alarm"] = False

            EAR_txt = f"EAR: {round(EAR, 2)}"
            DROWSY_TIME_txt = f"DROWSY:
{round(self.state_tracker['DROWSY_TIME'], 3)} Secs"
            plot_text(frame, EAR_txt, self.EAR_txt_pos,
self.state_tracker["COLOR"])
            plot_text(frame, DROWSY_TIME_txt, DROWSY_TIME_txt_pos,
self.state_tracker["COLOR"])

        else:
            self.state_tracker["start_time"] = time.perf_counter()
            self.state_tracker["DROWSY_TIME"] = 0.0
            self.state_tracker["COLOR"] = self.GREEN
            self.state_tracker["play_alarm"] = False

            # Flip the frame horizontally for a selfie-view display.
            frame = cv2.flip(frame, 1)

        return frame, self.state_tracker["play_alarm"]
```