**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“___”_____2024 р.

# Дипломна робота

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою**
***“Цифрові технології в енергетиці”***
**зі спеціальності *122 “Комп’ютерні науки”***

**на тему:** **Визначення напрямку зору на основі нейронної мережі**

Виконала:
студентка IV курсу, групи ТМ-02

*Казанішена Ірина Валеріївна* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник:
*д.т.н., професор Сергій ОТРОХ* _____
(посада, вчене звання ,науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент: *доцент кафедри інженерії програмного забезпечення в енергетиці, к.т.н., Валерій КУЗЬМІНИХ* _____
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль: *асистент Володимир РУДИК* _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2024

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

«__»_____2024 р.

**ЗАВДАННЯ**
**на дипломну роботу студенту**

**КАЗАНІШЕНІЙ Ірині Валеріївні**
(прізвище, ім’я, по батькові)

1. Тема роботи **Визначення напрямку зору на основі нейронної мережі**

керівник роботи ***Отрох Сергій Іванович, д.т.н., професор***
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 202__ р. №_____

2. Термін подання роботи студентом **07.06.2024 року**

3. Вихідні дані до роботи мова програмування Python, бібліотека OpenCV, NumPY, середовище розробки PyCharm,

4. Перелік питань, які потрібно розробити _____
Дослідити принципи визначення напрямку зору, використовуючи технології OpenCV, визначити потрібний метод для пошуку напрямку зору в реальному часі, розробити метод для визначення темних кругів, визначати дані для експорту до дата-сету, розробити алгоритм експорту даних.

5. Орієнтований перелік ілюстративного матеріалу: Актуальність; Мета роботи; Організація процесу розробки; Діаграма прецедентів; Діаграма компонентів; Метод перетворення Хафа; метод Гауса-Габора; Формування дата-сету; Засоби розробки; Користувацький інтерфейс; Висновки.
6. Дата видачі завдання «15» вересня 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Затвердження теми роботи | 17.05.24 | |
| 2. | Вивчення та аналіз задачі | 17-20.04.24 | |
| 3. | Розробка архітектури та загальної структури системи | 21-25.04.24 | |
| 4. | Розробка частин окремих підсистем | 25.04-02.05.24 | |
| 5. | Програмна реалізація системи | 03-07.05.24 | |
| 6. | Оформлення пояснювальної записки | 08-12.05.24 | |
| 7. | Захист програмного продукту | 13-17.05.23 | |
| 8. | Передзахист | 03-06.06.23 | |
| 9. | Подання готової роботи на кафедру | 07.06.2024 | |
| 10. | Захист | 17-21.06.2024 | |

Студент _____ **Ірина КАЗАНІШЕНА**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **Сергій ОТРОХ**
(підпис) (ім'я, ПРІЗВИЩЕ)

# АНОТАЦІЯ

Дана дипломна робота виконана на тему: «Визначення напрямку зору на основі нейронної мережі» студенткою кафедри цифрових технологій в енергетиці ННІ ІАТЕ Казанішеною Іриною Валеріївною зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Цифрові технології в енергетиці». Дана робота демонструє усі теоретичні та практичні частини створення програмного продукту для визначення напрямку зору на основі нейронної мережі в реальному часі. Дипломна робота складається зі вступу, загальних висновків, списку використаних джерел та містить в собі 5 таких основних розділів, як: «Задача пошуку напрямку зору на основі нейронної мережі», «Алгоритми роботи програми», Вибір засобів розробки», «Розробка програмного забезпечення» та «Робота користувача з програмною системою». Загалом робота займає 53 сторінки, містить 13 ілюстрацій, 2 додатки та 11 джерел в переліку посилань.

**Метою даної роботи** є створення надійного програмного забезпечення для пошуку та визначення напрямку зору на основі нейронної мережі у реальному часі.

**Актуальність теми.** Робота з дрібними деталями, мікросхемами, годинниковими механізмами, ювелірними прикрасами, артеріями та кровоносними судинами – все це потребує дуже уважної та кропіткої роботи без помилок, тому окуляри, які будуть в реальному часі визначати напрямок зору людини та направляти туди влаштовані світлові промені, значно спростять роботу вищезазначених фахівців та покращать якість їх роботи.

**Методи та засоби:** метод перетворення Хафа , перетворення Гауса-Габора, Гаусове згладжування, фільтр Габора, мова програмування Python, бібліотека комп'ютерного зору OpenCV, бібліотека для швидкого обчислення складних операцій NumPy.

**Результатом** є програмне забезпечення, для визначення напрямку зору на основі нейронної мережі в реальному часі.

**Ключові слова**: OPENCV, МЕТОД ХАФА, НАПРЯМОК ЗОРУ, PYTHON.

# ABSTRACT

This graduate work was written on the topic: “Determining the direction of vision based on a neural network” by a student of the Department of Digital Technologies in Energy of the Educational and Research Institute of Nuclear and Thermal Energy Kazanishena Iryna Valeriivna, specialty 122 “Computer Science” under the educational and professional program “Digital Technologies in Energy”. This work demonstrates all the theoretical and practical parts of creating a software product for determining the direction of vision based on a neural network in real time. The graduate work consists of an introduction, general conclusions, a list of references and includes 5 main sections, such as: “The task of finding the direction of vision based on a neural network”, “Algorithms of the program”, “Selection of development tools”, “Software development” and “User's work with the software system”. In total, the work occupies 53 pages, contains 13 illustrations, 2 appendixes and 11 sources in the list of references.

**Relevance of the topic.** Working with small parts, microcircuits, clockwork, jewelry, arteries and blood vessels - all this requires very careful and painstaking work without errors, so glasses that will determine the direction of a person's vision in real time and direct arranged light rays there will greatly simplify the work of the above specialists and improve the quality of their work.

**Methods and tools:** Hough transform method, Gaussian-Gabor transform, Gaussian smoothing, Gabor filter, Python programming language, OpenCV computer vision library, NumPy library for fast calculation of complex operations.

**The result** is a software for determining the direction of vision based on a neural network in real time.

**Keywords:** OPENCV, HAFA METHOD, DIRECTION DETECTION, PYTHON.

# ЗМІСТ

# СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ТЗ (технічне завдання) – визначені завдання, які потрібно виконати, поставлена задача.

Дата-сет (Data set) – це набір даних, організований для аналізу або навчання моделей машинного навчання.

Тайм-менеджент (Time management) – це практика організації та планування часу для підвищення ефективності та продуктивності.

Дедлайн (від англ. deadline) – крайній термін (дата або час).

Юзер (user – користувач) – людина, що користується програмним продуктом, та його функціями.

БД (база даних) – це організована колекція даних, яка зберігається та управляється за допомогою системи керування базами даних.

R-TOD (Real-Time Object Detection) – це процес виявлення та класифікації об'єктів на зображеннях або відео у реальному часі за допомогою алгоритмів комп'ютерного зору та машинного навчання.

HSV (також HSB) – колірна модель, заснована на трьох характеристиках кольору: колірному тоні (Hue), насиченості (Saturation) і значенні кольору (Value), який також називають яскравістю (Brightness).

Eye-tracking (технологія відстеження погляду) – це процес відстеження рухів очей людини для аналізу їх поведінки або взаємодії з пристроями, дисплеями або програмним забезпеченням.

# ВСТУП

У сучасному світі комп'ютерні технології та штучний інтелект знаходяться на передовому рівні розвитку, покращуючи нашу повсякденну діяльність у взаємодії з технологіями.

Сучасні досягнення в області штучного інтелекту та машинного навчання трансформують підходи до вирішення складних завдань, зокрема тих, що стосуються аналізу зображень та відео. Одним із напрямків, який набуває все більшого значення, є розробка систем відстеження напрямку зору. Це має велике застосування в різних сферах, починаючи від побутового застосування та закінчуючи медичними дослідженнями. Ця задача є надзвичайно важливою для багатьох прикладних галузей, зокрема для систем безпеки, медичних діагностичних систем, інтерфейсів взаємодії людина-комп'ютер, а також віртуальної та доповненої реальності. Технологія, що дозволяє точно визначити напрямок погляду людини, відкриває нові можливості для підвищення безпеки, ефективності та інтерактивності систем, що використовуються в різних сферах.

Одним із ключових методів досягнення цієї мети є використання нейронних мереж, які можуть вивчати складні залежності між вхідними даними та вихідними результатами. У наш час розвиток нейронних мереж кардинально змінив підходи до вирішення задачі визначення напрямку зору. Нейронні мережі демонструють високі результати у розпізнаванні зображень, обробці відео та інших складних задачах, завдяки чому стали ключовим інструментом у цій галузі.

У цьому контексті нейронні мережі виступають як потужний інструмент для аналізу та інтерпретації візуальних даних. Нейронна мережа - це обчислювальна модель, натхненна біологічними нейронними системами, яка складається з вузлів (нейронів) та з'єднань між ними. Ці мережі здатні навчатися на основі прикладів і виявляти складні патерни у великих наборах даних. Завдяки своїй здатності до навчання та адаптації, нейронні мережі демонструють виняткову точність у

розпізнаванні та класифікації зображень. Ці переваги роблять їх ідеальними для застосування у завданні визначення напрямку зору, де висока точність та надійність є критично важливими.

Поставлена задача є дуже актуальною в наш час та, на жаль, я не знайшла подібних алгоритмів, де можна було б коректно визначати напрямок. Також не було знайдено алгоритму, який визначає напрям у реальному часі, а це є невід’ємною та однією із ключових частин розробки.

Для реалізації системи визначення напрямку зору на основі нейронних мереж використовуються різні бібліотеки Python, серед яких особливо виділяються OpenCV та NumPy. OpenCV – це потужна бібліотека з відкритим вихідним кодом, призначена для комп'ютерного зору і машинного навчання. Вона забезпечує інструменти для обробки зображень та відео, включаючи фільтрацію, трансформації, виявлення об'єктів і багато іншого. NumPy, у свою чергу, є бібліотекою для роботи з масивами і матрицями, що надає високопродуктивні функції для чисельних обчислень.

Комбінація OpenCV та NumPy дозволяє ефективно обробляти та аналізувати візуальні дані, що є критично важливим для навчання та використання нейронних мереж у задачі визначення напрямку зору. Використання цих бібліотек забезпечує необхідну гнучкість та продуктивність, дозволяючи швидко прототипувати та розгортати складні моделі комп'ютерного зору.

У цьому дипломному проекті метою є розробка та реалізація системи, яка здатна визначати напрямок зору людини на основі зображень, отриманих з камери. Ця дипломна робота присвячена дослідженню та розробці методів визначення напрямку зору на основі нейронних мереж. У роботі будуть розглянуті сучасні підходи до цієї проблеми, проаналізовані їхні переваги та недоліки, а також запропоновані нові методики, що можуть підвищити ефективність вирішення задачі.

Для чіткого та успішного досягнення поставленої мети було поставлено

такі завдання:

1) ознайомитись та детально дослідити алгоритм для пошуку напрямку зору на основі нейронної мережі на зображенні в реальному часі;

2) розробити програмний код для успішного та правильного знаходження напрямку зору, використовуючи обрані методи та бібліотеки для визначення напрямку зору в реальному часі;

3) сформувати базу даних, в якій будуть зберігатись усі вихідні значення результати програми.

За підсумком виконання роботи було подано заявку на створення авторського свідоцтва на програмний код.

Враховуючи великий потенціал застосування таких систем у сферах від автомеханіки до розвитку допоміжних технологій в області нейрохірургічної медицини, дослідження в даній області є актуальним та перспективним.

У цьому контексті, в даному дипломному проекті буде розглянуто та детально проаналізовано різноманітні методи та підходи до визначення напрямку зору на основі нейронних мереж. Дослідження, яке проводилось в рамках проекту, спрямоване на розробку ефективного та точного алгоритму, здатного працювати в реальному часі та в різних умовах.

Дипломна робота складається з 5 основних розділів, в яких детально описано та обґрунтовано поставлені задачі для успішної розробки даного програмного продукту, обрані алгоритми роботи програми, використані засоби розробки, програмні частини коду серверної частини розробки, інтерфейс програми, функціонал та взаємодія з користувачем. Записка налічує 52 сторінки та містить 13 ілюстрацій, 1 додаток та 11 використаних джерел.

Таким чином, цей дипломний проект є спробою внести вагому інновацію у сферу розробки систем комп'ютерного зору та штучного інтелекту, а також відкрити нові перспективи в їхньому застосуванні.

# 1 ЗАДАЧА ПОШУКУ НАПРЯМКУ ЗОРУ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

Метою даної роботи є створення прикладного програмного продукту для пошуку напрямку зору на основі нейронної мережі та занесення вихідних даних у базу даних для подальшого навчання нейронної мережі.

Розробку поставленої задачі можна класифікувати на декілька основних частин роботи:

– аналіз поставленої задачі, розробка ТЗ та допоміжних діаграм;

– аналіз існуючих програмних рішень, аналіз предметної області;

– визначення алгоритмів роботи для пошуку напрямку зору на основі нейронної мережі;

– розробка програмного продукту та його тестування.

В результаті усіх вищеперелічених етапів, які будуть успішно виконані, можливо швидко та точно виконати усі частини поставленої задачі.

## 1.1 Аналіз поставленої задачі

Тему дипломної роботи можна сформулювати так: «Визначення напрямку зору на основі нейронної мережі».

Аналізуючи тему бакалаврської роботи, було прийнято рішення розробити алгоритм, який буде визначати темні кругові ділянки на зображенні, використовуючи засоби нейронної мережі. В результаті роботи даного алгоритму буде отримано центр зіниці ока та координати її розміщення.

Мій програмний продукт буде обраховувати координати центра зіниці ока в реальному часі із використанням бібліотеки комп’ютерного зору. Дана технологія відома в технічному колі, як R-TOD.

**Real-Time Object Detection** (виявлення об’єктів у реальному часі) – це завдання комп’ютерного зору, яке передбачає ідентифікацію та визначення місцезнаходження цікавих об’єктів у відеопослідовності в реальному часі за допомогою швидкого висновку, зберігаючи при цьому базовий рівень точності [1].

Зазвичай це вирішується за допомогою алгоритмів, які поєднують методи виявлення об’єктів і відстеження для точного виявлення та відстеження об’єктів у реальному часі. Вони використовують комбінацію виділення ознак, створення пропозицій об’єктів і класифікації для виявлення та локалізації об’єктів інтересу.

Отже, моя програма використовує камеру девайсу, з якої сканується пошуковий елемент – зіниця ока. Для демонстрації знайденого об’єкту на екран виводиться ваша камера зі знайденим колом та його центром. Також показуються координати X та Y, які являють собою точку центру зіниці ока. Щоб продемонструвати зміну напрямку зору між координатами попереднього центру зіниці та поточної координати, програма малює лінію.

Також додатково, окрім основного вікна із зображенням камери, будуть представлені ще 2 вікна, такі як: вікно із контуром об’єктів та вікно з маскою чорних об’єктів. Алгоритм повинен використовувати певний діапазон темного кольору.

Отже, буде отримано програму, яка матиме цілком виправдане підґрунтя для роботи. Пошук зіниці ока буде здійснений, використовуючи OpenCV, таким чином будуть використовуватись технології штучного інтелекту, що задовольняє умови поставленої задачі.

Для закріплення використання нейронних мереж – я запропонувала створити базу даних, в якій буде зберігатись центр об’єкта, зображення до знаходження, зображення контуру, зображення маски та зображення знайденого кола. Ця база даних буде використовуватись для навчання згорткової нейронної мережі.

## 1.2 Організація процесу виконання задачі

Для успішної розробки програмного продукту було розділено усі задачі на підзадачі, які будуть послідовно виконуватись до кінцевого результату – робочий алгоритм для визначення напрямку зору, який у реальному часі буде визначати центр кола зіниці та його координати.

Кожна поставлена задача повинна мати чіткий дедлайн. Завдяки правильному тайм-менедженту можна вчасно виконати усі задачі у визначений термін.

**Time management** (тайм-менеджмент, керування часом) – це процес організації поставлених задач за допомогою сукупності різних методів управління часом. Він полягає у встановленні часових рамок для виконання пріоритетних завдань [2].

Також потрібно розділити основні частини роботи на підзадачі по пріоритетності для успішного виконання задачі.

### 1.2.1 Декомпозиція поставленої задачі

Поставлену задачу дуже важливо розбити на послідовні етапи розробки, щоб унеможливити розсіювання у роботі та не проґавити важливі деталі розробки. Це забезпечить якісне сконцентроване виконання усіх визначених підзадач. Крім того, цей підхід дозволяє дипломному керівникові чітко розуміти, на якому етапі розробки знаходиться студент.

**Декомпозиція** – розподіл великої задачі на менші складові, які покрокового виконуються для досягнення поставленої цілі [2].

Декомпозиція задачі визначення напрямку зору буде визначатись так:

– аналіз поставленої задачі;

– пошук та аналіз існуючих сервісів;

– визначення переліку функцій системи;

– створення списку задач для технічного завдання;

– затвердження технічного завдання у дипломного керівника;

– оцінка часу на розробку;

– створення допоміжних діаграм;

– визначення алгоритмів роботи застосунку;

– створення схематичного інтерфейсу;

– кодування підпрограми «Визначення чорних кругів»;

– кодування екранів для зображення контуру та маски;

– визначення точки центру знайденої зіниці;

– створення БД для дата-сету;

– кодування механізму експорту даних до БД;

– перевірка функціональності, тестування;

– виправлення недоліків та помилок.

В результаті виконання усіх наведених пунктів – я отримую успішно виконаний програмний продукт.

### 1.2.2 Створення діаграми Ганта

Після визначення декомпозиції ми можемо переходити до наступного етапу – планування усіх послідовних дій із урахуванням визначених часових меж. Для цього спочатку визначимо, які часові проміжки задані для певних задач.

Крайньою початковою точкою є 15 квітня 2024 року. Кінцевою точкою є 19 травня. У цей часовий проміжок було успішно пройдено переддипломну практику, під час якої і було розроблено даний програмний продукт.

Усі визначені раніше підзадачі було перенесено в тайм-менеджмент діаграму для того, щоб був можливий нагляд за поетапним виконанням визначеної задачі у

чіткий дедлайн та для розуміння процесу виконання певних підзадач. Time managment діаграмою виступає діаграма Ганта.

**Діаграма Ганта** (стрічкова діаграма, графік Ганта) – це інструмент для візуалізації графіка виконання проєктних завдань. Основною умовою створення цієї діаграми є розбиття великої задачі на менші компоненти та планування їх виконання протягом певних відрізків часу [2].

Для успішного управління проектами та їх планування діаграма Ганта використовується, як основний засіб. Графік Ганта було створено у Microsoft Visio. Ця програма виявилась дуже комфортною у користуванні та зрозумілою для проектування таких діаграм.

Щоб правильно організувати процес виконання роботи було враховано вихідні дні. Це можна визначити, проаналізувавши діаграму та зауваживши, що вона містить сірі стовпчики, які визначають вихідні дні, тобто суботу та неділю.

Після ознайомлення з діаграмою наглядно видно усі пункти, які нас цікавлять для успішного виконання поставленої задачі. У графіку Ганта присутня перша колонка із «назвою підзадачі», а в наступних прописані часові межі для виконання певних задач, тобто «початок» та «кінець».

У верхній частині діаграми знаходиться календар для зручного орієнтування в дедлайнах та також за допомогою календаря можна чітко зрозуміти, як саме проходять певні процеси: чи проходять вони послідовно, чи якісь певні задачі можуть проходити паралельно один із одним.

Враховуючи те, що усі задачі у мене будуть виконуватись послідовно одна за одною, то діаграма тайм-менеджменту буде містити лише послідовні процеси без залучення паралельних.

Адже кожен з пунктів пов’язаний один з одним та успішне виконання кожного з них залежить від результатів попереднього (рисунок 1.1).

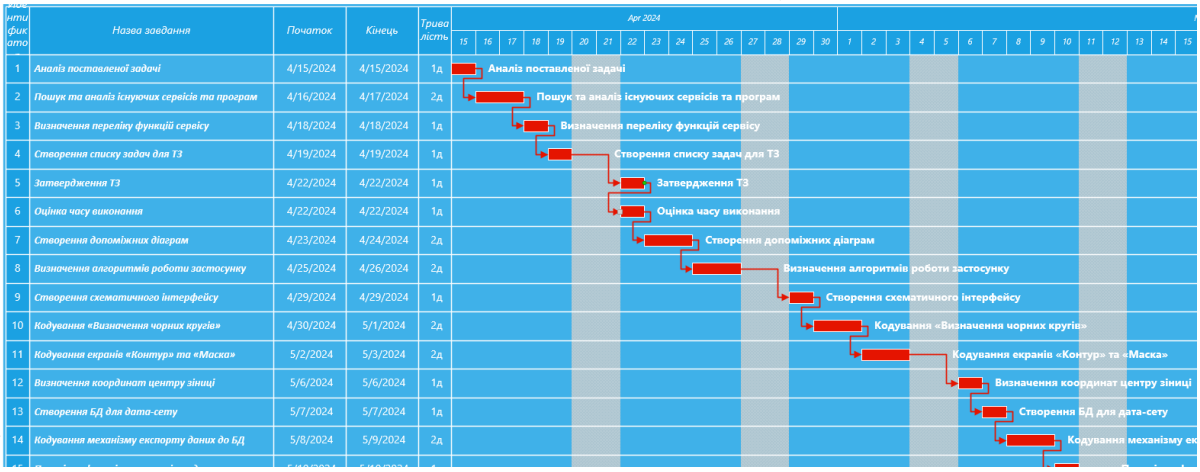



Рисунок 1.1 – Діаграма Ганта завдання 1-11

В першій частині діаграми ми бачимо послідовно виконані процесі для підготовки до розробки програмного продукту, які включають в себе: аналіз поставленої задачі, розробка необхідних діаграм, формування тз, пошук найкращих рішень для визначення зіниці та інші. Наступна частина діаграми більше показує розробку самого програмного продукту та підготовку до захисту (рисунок 1.2).


Рисунок 1.2 – Діаграма Ганта завдання 11-18

Отож, після детального аналізу створеної діаграми Ганта можна ознайомитись із усіма поставленими підзадачами, кожна з яких має певні часові межі, які також зазначені у графіку, що дає змого дотриматися усіх дедлайнів.

### 1.3 Опис основних задач програмного продукту

Головною ціллю моєї розробки я виділила для себе – створення такого алгоритму, який буде працювати у реальному часі, використовуючи лише найновіші технології, аби забезпечити швидку та чітку роботу програмного продукту та мала високий рівень адаптації до різних засобів.

Мій програмний продукт забезпечений такими задачами, як:

– виведення екрану з зображенням контурів чорних об’єктів;

– виведення екрану з зображенням маски чорних об’єктів;

– пошук кругів серед контурів чорних об’єктів методом Хафа;

– демонстрація центру знайденого круга та його координати;

– обмеження зони пошуку кругів;

– експорт успішних результатів до бази даних;

– надання можливості користувачу змінювати обмежувальну зону для пошуку.

Тому враховуючи усі попередньо описані задачі, можна впевнено визначити, що створений алгоритм буде чітко та правильно виконувати поставлену ціль. Також програма буде надійною і вираховувати значення у реальному часі, вивівши опісля отримані результати на екран для демонстрації їх користувачеві.

## 1.4 Призначення продукту та його користувачі

Основною задачею є розробка програмного коду, який в свою чергу є декомпозицією масштабнішого проєкту. Тому, враховуючи цей факт, ми можемо визначити, що призначення створеного продукту із визначенням напрямку зору буде ідентичним із призначенням фінального механізму цієї масштабної розробки.

Ідея самого продукту полягає у створенні окулярів, які будуть містити функцію зі зчитуванням напрямку зору та влаштовану адаптацію променів світла у точку, куди дивиться користувач у реальному часі. Таким чином, дана розробка знайде своє покликання та буде корисною у багатьох сферах, а саме: у сфері медицини, приладобудування, автомеханіки, ювелірних виробів тощо. Вагомим та значущим прикладом використання цього механізму у сучасних реаліях є застосування його медиками у прифронтовій зоні. Адже усі ми розуміємо, що у прифронтових зонах медики мають обмежені умови для роботи, на відмінну від працівників у медичних закладах. Тому у випадку, коли у прифронтового медика стоїть задача, негайно оглянути пораненого чи потрібно провести термінову операцію військовому з критичними травмами, які не дають додаткового часу, аби довезти військового до прифронтового шпиталю, яке забезпечене потрібним обладнанням, яке передбачає якісне освітлення. Тому ці окуляри дадуть змогу медикам провести усі необхідні дії в негайному порядку на місці, щоб врятувати життя людей і не стати ціллю для ворога.

Користувачами даного продукту можуть стати медики, ювеліри, автомеханіки, годинникарі, стоматологи, інженер-конструктори та інші. Таким чином можна розглянути певний приклад застосування розробки для працівника, що займається ремонтом і проектуванням електронних пристроїв, роботою з мікросхемами та дрібними електронними компонентами. Даний фахівець працює з дрібними деталями, з мікросхемами, які потребують точного та чіткого спаювання, а для цього працівнику необхідне точкове світло у місце роботи в даний момент.

Як висновок, можна підсумувати та визначити, що дана розробка програмного забезпечення знайде своє важливе та вкрай необхідне застосування в сучасному світі для багатьох сфер діяльності, які цього потребують. Також цей продукт зможе не лише полегшити та пришвидшити роботу фахівців, а й врятувати життя людей, що є неймовірно важливим аспектом цієї ідеї розробки.

## 1.5 Опис предметної області

Предметна область для визначення напрямку зору в реальному часі охоплює наукові дослідження та розробки, які спрямовані на розуміння, вимірювання та використання напрямку зору людини для потрібних цілей.

Моя розробка вдало підходить під таку предметну область, як технології відстеження погляду (eye-tracking).

**Eye-tracking (технологія відстеження погляду)** – це технологія, яка дозволяє вимірювати та аналізувати напрямок і рух очей людини, також відстежувати лінії погляду або точки погляду людини. Дана технологія може використовуватись для збору даних, тобто про те, куди дивиться людина, як довго вона зосереджує свій погляд на певних об'єктах і як змінюється її погляд під час різних активностей [1].

Таку технологію відстеження погляду eye-tracking ще деколи можуть називати «окулографія». Ця технологія є цілком новою, тому в даний момент в Україні eye-tracking ще не отримала достатнього поширення, але загалом за останні роки можна визначити, що розвиток цієї технології зійшов з мертвої точки.

Тобто можна зрозуміти, що даний продукт подібний до розробок, які підлягають вищезазначеній предметній області.

В дану предметну область входять такі сфери застосування:

– наукові дослідження: вивчення когнітивних процесів, психологічних реакцій та поведінки людей;

– маркетинг і UX-дослідження: аналіз взаємодії користувачів з вебсайтами, рекламними матеріалами та продуктами для покращення дизайну та ефективності;

– медицина: діагностика та лікування неврологічних розладів, реабілітація пацієнтів з порушеннями зору;

– віртуальна та доповнена реальність (VR/AR): створення інтуїтивно зрозумілих інтерфейсів, що реагують на погляд користувача;

– автомобільна індустрія: моніторинг уваги водія для підвищення безпеки та впровадження систем допомоги водієві;

– ігри та розваги: інтерактивні ігри та програми, що реагують на погляд користувача.

Задавши собі важливе питання, чи існує загалом в даній предметній області справді подібна розробка, то було визначено, що в даний час у просторах інтернету не знайдено повноцінного механізму, який буде направляти промені світла у точку, куди дивиться користувач у реальному часі.

## 1.6 Розробка допоміжних діаграм

Щоб розробка програмного забезпечення була успішною та чітко і правильно виконаною обов’язково потрібно зайнятись створенням допоміжних діаграм, які в процесі роботи зможуть зекономити час та допоможуть уникнути втрати важливих деталей для реалізації. За допомогою таких діаграм можна визначити загальну логіку роботи програмного продукту та допоможуть визначити основні задачі та процеси, які потрібно розробити.

Тому для своєї успішної розробки програми, перед написанням самого програмного коду, я визначила такі важливі діаграми, які мені потрібно створити: діаграма прецедентів та діаграма компонентів.

Завдяки створеним діаграмам я зможу наглядно розуміти, як буде працювати програмний продукт та які задачі він буде виконувати та звісно важливою

перевагою наявності цих допоміжних діаграм буде те, що, проаналізувавши їх, мій дипломний керівник та комісія зможуть правильно та чітко зрозуміти суть моєї розробки. Та також в цей самий час дані діаграми дуже допоможуть мені при створенні запланованих алгоритмів для чіткої роботи мого програмного продукту, включаючи усі деталі, які я визначила на початку роботи.

### 1.6.1 Опис діаграми прецедентів

Щоб наглядно пояснити та показати вимоги до розробленого продукту та його коректної роботи, було реалізовано діаграму прецедентів.

**Діаграма прецедентів** (Use-case діаграма) – це діаграма, яка демонструє співвідношення та взаємодію акторів і прецедентів. Запити на функціонал відображені у вигляді прецедентів, також use-case діаграма створюється за допомогою мови UML [3].

**UML** (Unified Modeling Language) – це мова, яка використовується у програмуванні, зокрема об’єктно-орієнтованому та є уніфікованою мовою моделювання. Вона є стандартом для створення абстрактних моделей систем в об'єктно-орієнтованому програмуванні та є вагомою частиною будь-якого процесу розробки програми. За допомогою мови UML можуть створюватись допоміжні діаграми, так звані UML-моделі, тобто діаграма станів, діаграма компонентів тощо [3].

Use-case діаграма може показати та пояснити усі функціональні можливості, що мають виконуватись на конкретні запити користувача завдяки системі. «Система» і «Користувач» є головними акторами цієї діаграми (рисунок 1.3).


Рисунок 1.3 – Діаграма прецедентів

Діаграму прецедентів було створено у Microsoft Visio. У діаграмі показані усі функції, які належать системі та також можна побачити наглядно, які саме запити може виконувати юзер.

Основною частиною зв’язку користувача з програмною системою є натиск кнопки для калібрування – це запускає механізм пошуку зіниці користувача та адаптує світлові промені. Для того, щоб успішно пройти калібрування, користувач має сфокусувати свій погляд на конкретному предметі. Після цього система адаптується до погляду юзера та починає слідкування за напрямком зору.

Актор «Система» виконує чотири основні функції: слідкування за напрямком зору, калібрування, визначення зіниці ока та поворот променів світла за напрямком зору.

Визначення зіниці ока включає в себе визначення кругів методом Хафа та визначення координат центру зіниці ока. Цей прецедент є частиною прецеденту з калібруванням.

В прецеденті експорт знайдених значень до дата-сету передаються успішні результати знайденої зіниці ока до бази даних для подальшого навчання згорткової нейронної мережі.

Серед переданих результатів будуть такі дані, як:

– центр координати зіниці;

– зображення кругу довкола зіниці;

– зображення ока;

– маска об’єкта;

– знайдені круги методом Хафа.

Отож, розробивши допоміжну діаграму прецедентів, можна чітко визначати функціональні можливості системи при певних запитах користувача та його потребах.

### 1.6.2 Розробка діаграми компонентів

Також було створено діаграму компонентів, на якій можна наглядно побачити основні елементи програмного продукту.

**Діаграма компонентів** – це тип діаграми в UML (Unified Modeling Language), що демонструє організацію та певні залежності між фізичними програмними компонентами системи. Вона використовується для моделювання фізичної реалізації системи, включаючи програмні компоненти, бібліотеки, файли, виконувані модулі та інші фізичні артефакти. Діаграма компонентів допомагає зрозуміти, як різні частини системи взаємодіють та залежать одна від одної [3].

Дану діаграму було створено за допомогою веб-застосунку draw.io, використовуючи посилання https://app.diagrams.net/. Завдяки зручному функціоналу та інтерфейсу створення діаграми компонентів пройшло доволі швидко та цілком успішно, а окрім цього, ще й без додаткових витрат. Завдяки

цьому сервісу я змогла створити якісну візуалізацію своєї розробки та частин мого продукту (рисунок 1.4).

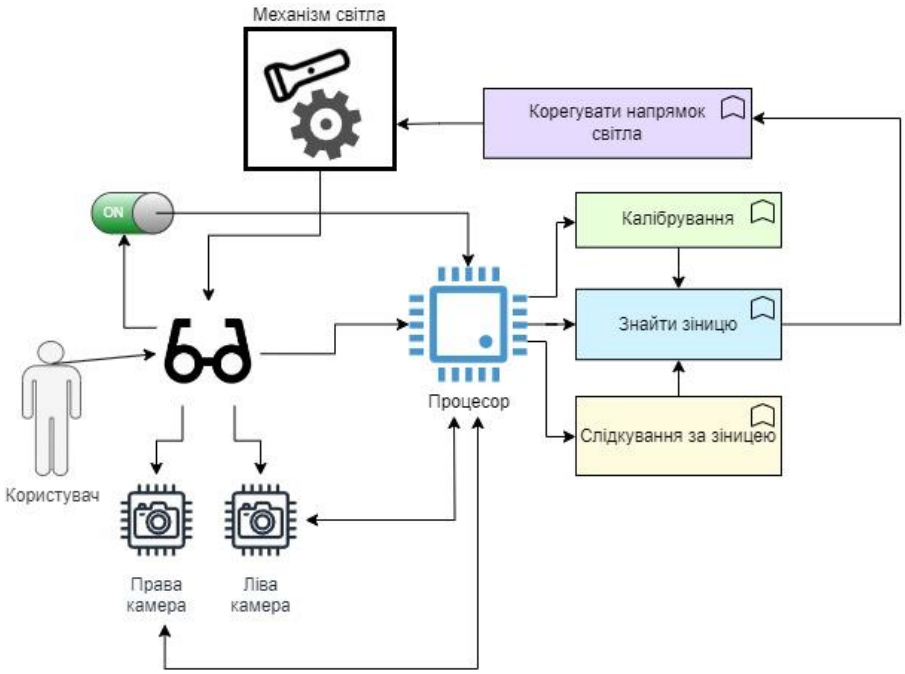



Рисунок 1.4 – Діаграма компонентів

У цій діаграмі можна наглядно побачити усі взаємозв’язки та загальну послідовність роботи усього функціоналу

На діаграмі зображено користувача, який взаємодіє з окулярами, запускаючи їх функціонал за допомогою певної кнопки. В цих окулярах влаштовані дві камери, які чітко зчитують очі користувача в реальному часі. Процесор обраховує напрямок зору юзера та зручно підлаштовує механізм світла до напрямку зору.

Існують також функціонали з калібруванням, для цього користувачу потрібно сфокусувати свій зір на якомусь предметі. І тоді після калібрування система знову слідкує за напрямком зору користувача.

# 2 АЛГОРИТМИ РОБОТИ ПРОГРАМИ

У цьому розділі я опишу та продемонструю усі алгоритми роботи програми, методи, які були використані мною для успішного виконання певних елементів поставленої задачі. Можна виділити основні та найважливіші методи, які будуть описані далі у цьому розділі: метод Хафа та метод Гауса-Габора.

Для успішної розробки програмного продукту використовувались далі описані та продемонстровані методи.

## 2.1 Метод перетворення Хафа

Для основної задачі – визначення напрямку зору потрібно розробити алгоритм, що буде визначати круги в реальному часі. В даній роботі було обрано один із основних методів – метод перетворення Хафа. Метод перетворення Хафа є потужним інструментом в обробці зображень і знаходить широке застосування у різних галузях, де важливо точно виявити геометричні форми на зображеннях. По даному методі було знайдено досить багато інформації для вивчення, саме тому було обрано саме цей метод.

**Метод перетворення Хафа** (Hough Transform) – це методика, яка використовується для виявлення об'єктів на зображеннях за допомогою перетворення простору зображення в простір параметрів. Тому це дозволяє знаходити об'єкти з певними формами, такими як прямі лінії, кола або інші прості геометричні фігури, незважаючи на присутність шуму та неповні дані [4].

Алгоритм перетворення Хафа для прямих базується на аналізі двох параметрів, які і визначають пряму. В такому випадку простір для перетворення буде мати два виміри та кожна точка у цьому вимірі слугуватиме накопичувачем для визначення та виявлення прямої, яка буде описуватись таким рівнянням:

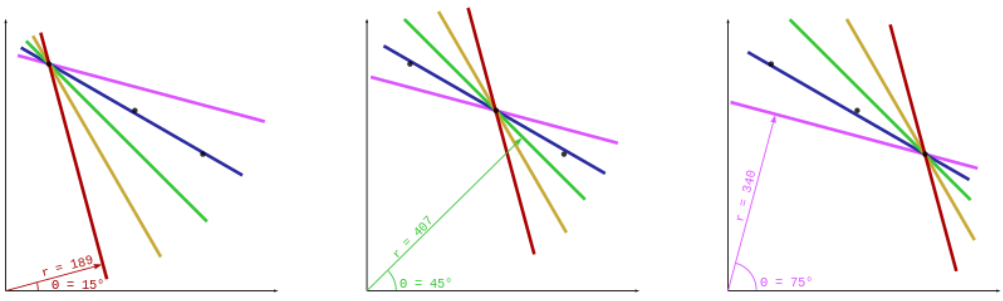
$$r = x\cos\theta + y\sin\theta.$$

Кожна точка, яка буде виявлена на контурах даного зображення вносить свої дані у накопичувач.

Розмірність накопичувача дорівнює значенню невідомих параметрів, тобто двом, оскільки ми розглядаємо квантовані значення r та θ у парі (r, θ). Для кожного пікселя в (x, y) та його оточенні алгоритм перетворення Хафа визначає, чи міститься достатньо свідчень про пряму в даному пікселі. Якщо значень достатньо, то він обчислює параметри (r, θ) цієї прямої, а потім проводить пошук засіку накопичувача, до якого вони потрапляють, і збільшує значення цього засіку.

Зазвичай шляхом пошуку локальних максимумів у накопичувальному просторі, що відбувається через знаходження засіків з найвищими значеннями, можна виділяти найбільш ймовірні прямі та отримувати їхні наближені геометричні параметри.

Кінцевий результат перетворення Хафа для прямих - це двовимірний масив або матриця, яка схожа на накопичувач. Один із вимірів цієї матриці квантується за кутом θ, а інший - за відстанню r. Кожен елемент цієї матриці містить значення, що представляє суму точок або пікселів, що знаходяться на прямій, яка описана квантованими параметрами (r, θ). Таким чином, елемент з найвищим значенням вказує на пряму, яка найбільш представлена на вхідному зображенні (рисунок 2.1).




| θ | r |
|---|---|
| 15 | 189.0 |
| 30 | 282.0 |
| 45 | 355.7 |
| 60 | 407.3 |
| 75 | 429.4 |

| θ | r |
|---|---|
| 15 | 318.5 |
| 30 | 376.8 |
| 45 | 407.3 |
| 60 | 409.8 |
| 75 | 385.3 |

| θ | r |
|---|---|
| 15 | 419.0 |
| 30 | 443.6 |
| 45 | 438.4 |
| 60 | 402.9 |
| 75 | 340.1 |

Рисунок 2.1 – Перетворення Хафа

Як приклад розгляньмо три точки даних, показані тут як чорні крапки:

– крізь кожну точку даних проведено кілька прямих під різними кутами, які відображено різними кольорами;

– під час перетворення Хафа внески від усіх пікселів у виявленому контурі накопичуються. Кожна пряма має свій опорний відрізок, який перпендикулярний до неї і перетинається з початком координат; в кожному випадку один з цих відрізків показано стрілкою;

– обчислюють довжину (тобто перпендикулярну відстань до початку координат) і кут кожного опорного відрізка; довжини та кути представлені в таблиці під діаграмами.

Після таких обчислень можна зробити висновок, що в усіх випадках опорний відрізок під кутом 60° має схожу довжину. Тому і зрозуміло, що відповідні прямі (сині на рисунку 2.1) дуже схожі. Таким чином, можемо вважати, що всі ці точки лежать близько до синьої прямої.

Первинно перетворення Хафа використовувалося лише для виявлення прямих на зображенні, однак пізніше його розширили для визначення положень будь-яких фігур, зазвичай – кола або еліпсів.

Зробити зміни у алгоритмі, щоб стало можливим виявлення колових фігур замість прямих ліній відносно легко:

– створюємо для початку накопичувальний простір, який буде містити для кожного пікселя певні комірки; початкове значення кожної комірки буде 0;

– для кожної контурної точки (i, j) на зображенні потрібно збільшити всі існуючі комірки, які відповідно до відомого рівняння кола $(i-a)^2+(j-b)^2=r^2$ можуть бути центрами кола; ці комірки у рівнянні будуть позначатись літерою $a$;

– знаходимо всі можливі значення $b$, що будуть задовольняти рівняння, для кожного можливого значення $a$, яке було знайдено попередньо;

– знаходимо локальні максимуми у накопичувальному просторі; ці комірки визначають кола, які було виявлено алгоритмом.

Метод перетворення Хафа слугує потужним та надійним засобом у обробці зображень та знаходить часте використання у багатьох галузях, де важливо саме точно і чітко виявити геометричні форми на певних зображеннях.

## 2.2 Перетворення Гауса-Габора

Перетворення Гаусса-Габора, відоме також як Габорове перетворення, поєднує в собі дві операції обробки зображень: Гаусове згладжування і фільтр Габора.

Далі будуть надані детальні пояснення обох цих операцій.

### 2.2.1 Гаусове згладжування

Для кращого зчитування зображення було використано метод Гаусового згладжування для зменшення зайвих деталей та шуму.

**Гаусове згладжування** (Розмивання Гауса, Gaussian Smoothing) – це процес, в основі якого використовується гаусівське ядро (гаусівська функція) для зменшення шуму та видалення деталей зображення. Ця операція допомагає зробити зображення більш розмитим, зменшити градієнти та виділити основні особливості [5].

Розмивання Гауса використовує функцію Гауса (яка також зустрічається в нормальному розподілі в області статистики) для розрахунку трансформації кожного пікселя у певному зображенні. Рівняння функції Гауса в одному вимірі має такий вигляд:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}.$$

Для прикладу роботи Гаусового згладжування ми використали зображення ока. Після обробки видно, як зникли шуми та зображення стало зручнішим для обробки методом Хафа (рисунок 2.2).

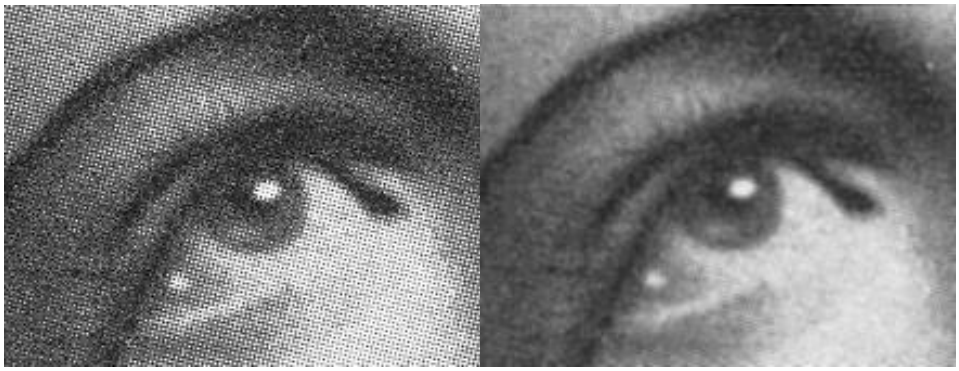


Рисунок 2.2 – Гаусове згладжування

Для двовимірного випадку, вираз складається з двох таких функцій, по одній для кожної осі виміру:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},$$

де x – це відстань від початку координат в осі абсцис,

y – це відстань від початку координат в осі ординат,

σ – стандартне відхилення розподілу Гауса.

У випадку, коли метод розмивання має застосовуватись у двох, буде отримана поверхня, що має такі контури, як концентричні кола розподілу Гауса з центральної точки.

Показники з цього розподілу будуть використані для формування матриці згортки. Кожне нове значення пікселя обчислюється, як середнє зважене в околі пікселя. Оригінальне значення поточного пікселя має вищу вагу, тобто матиме максимальний показник розподілу Гауса, а значення сусідніх пікселів отримуватимуть все меншу вагу залежно від їх віддаленості від поточного пікселя.

Це і створює ефект тої розмитості, яким і обумовлена операція згладжування Гауса, цей ефект зберігає кордони та краї досконаліше, ніж інші існуючі аналоги фільтрів розмиття.

Розмивання Гауса може мати застосування у випадку з чорно-білим зображенням, щоб отримати його згладжене зображення з півтонного друку, тобто той, що найчастіше використовують у газетах, журналах та інших виданнях для надання глибини зображенням.

### 2.2.2 Фільтр Габора

Для покращення чіткості зображення та визначення головних елементів я використовую фільтр Габора.

**Фільтр Габора** (Gabor Filter) – це операція, що має використання у визначенні текстур та країв на зображенні. Фільтр Габора є спеціалізованим фільтром, який має як реальну, так і уявну частину. Фільтр визначає, наскільки зображення відповідає певним шаблонам або фільтрам, які розташовані на різних частотах і орієнтаціях [6].

Для прикладу застосування фільтра Габора я використала зображення відбитків пальця. Ми бачимо, як непомітні лінії стають чіткими та явними. Завдяки фільтру ми виявляємо основне із зображення (рисунок 2.3).

Рисунок 2.3 – Приклад відбитка пальця обробленого фільтром Габора

У процесі цифрової обробки зображень цей фільтр використовується для визначення меж об'єктів. Тому через властивість взаємозв'язку між згорткою в часовій області та множенням у частотній області, перетворення Фур'є імпульсної передавальної характеристики фільтра Габора представляє собою результат згортки перетворень Фур'є гармонійної функції та гауссіана.

Фільтри Габора тісно пов'язані з вейвлетами Габора, оскільки їх можна побудувати шляхом послідовного стиснення та обертання. Простір Габора (результат згортки фільтра з сигналом) часто застосовується у різних областях обробки зображень, зокрема, для визначення райдужної оболонки в біометричних системах безпеки та в автоматизованих системах контролю доступу на основі розпізнавання відбитків пальців.

Для побудови одновимірного фільтра Габора застосовується така формула:

$$G(x) = \exp(-\frac{x^2}{2\sigma^2})\cos(2\pi\theta x),$$

де σ – стандартне відхилення Гаусового ядра, що визначає амплітуду функції,

θ – частота коливань, що визначається як $\theta = \frac{1}{\mathrm{T}}$,

де T – період функції $\cos(2\pi\theta \mathrm{x})$.

Чим більше $\sigma$, тим більш пологою буде виглядати функція. Чим менше $\sigma$, тим більше гострий пік вийде в результаті побудови графіка функції відповідно.

Функція експоненти, яка була вищенаведена, має властивості нормального розподілу випадкової величини. Тому згідно з правилом трьох сигм, практично всі значення експоненти будуть знаходитись у інтервалі [-3 σ;3 σ]. Для аналізу сигналів значення функції розраховуються в саме таких зазначених межах.

Обробка одновимірного сигналу фільтром Габора проходить наступним чином:

Кожна точка вхідного сигналу F(x) перетворюється на відповідну точку вихідного сигналу $F'(x)$ і тоді шляхом усереднення значень вхідного сигналу F(t)

по області $t \in [x - \frac{n}{2}, x + \frac{n}{2}]$ та з урахуванням вагових коефіцієнтів $G(i)$ формули Габора:

$$F'(x) = \frac{1}{n}\sum_{i=1}^{n} F(x - \frac{n}{2} + i) * G(i),$$

де F(x) – вхідне значення сигналу в точці x,

$F'(x)$ – вихідне значення сигналу в точці x,

$G(i)$ – значення функції Габора, $i \in [0, n]$.

Особливості фільтрів Габора будуть такими:

– локалізація – Габорові фільтри здійснюють ефективну локалізацію, як у просторі, так і в частотній області, що дозволяє виявляти деталі на цілком різних масштабах та орієнтаціях;

– біологічна правдоподібність – функції Габора подібні до рецептивних полів нейронів зорової кори, що робить їх корисними для моделей зорового сприйняття.

Об'єднуючи ці дві операції, перетворення Гауса-Габора створює комплексний фільтр, що поєднує переваги обох методів. Цей підхід дозволяє одночасно позбавлятися від високочастотних складових для згладжування зображення та виділяти важливі текстурні деталі. Такий метод часто використовується у обробці зображень для виявлення об'єктів, визначення країв, а також для вирішення завдань у сферах комп'ютерного зору та обробки зображень.

# 3 ВИБІР ЗАСОБІВ РОЗРОБКИ

Для успішної розробки програмного забезпечення було обрано найоптимальніші та найактуальніші засоби розробки в сфері програмування. Обрані інструменти є фундаментом для створення подібних проектів за метою та функціоналом.

В цьому розділі далі я продемонструю та опишу, які саме засоби я використовувала при створенні програмного:

– мови програмування;

– інструменти та середовище розробки;

– бібліотеки для розробки продукту.

Окрім цього, варто також зазначити, які системні вимоги ставить перед собою розроблений програмний продукт та за яких умов він буде коректно та чітко працювати.

## 3.1 Використані мови програмування

Найпершим та найосновнішим етапом перед створенням програмного коду є вибір мови програмування, що стане надійним помічником у створенні бажаного продукту. Обрана мова програмування повинна задовольняти усі потреби розробки, бути зручною у використанні та актуальною в момент створення програми.

Тому свій вибір я одразу зробила в користь мови програмування Python. Ця мова зможе допомогти мені спростити кодування моєї основної апаратно-серверної частини, використовуючи вбудовані бібліотеки комп’ютерного зору, які вкрай необхідні для успішної реалізації мого проєкту. Також обраний мовний інструмент буде актуальним для подальшого масштабування моєї розробки, адже Python є

найпростішою мовою у застосуванні під час роботи зі згортковою нейронною мережею та зможе забезпечити якісний та коректний результат.

**Python (Пайтон)** – це скриптова мова програмування, що зараз є найпопулярнішою серед фахівців через свою інтерпретованість до об'єктно-орієнтованого програмування [7].

Ця мова має багато вагомих переваг, які допоможуть у створенні мого продукту, а саме:

– наявність потужного середовища розробки, яке є досить простим у використанні;

– можливість розв'язання математичних проблем різної складності, оперування цілими числами будь-якої величини;

– гнучкість та масштабованість;

– простий та лаконічний синтаксис, який скорочує час розробки програмного коду та спрощує її.

Тому з використанням саме цієї інтерпретованої мови програмування я зможу розробити алгоритм, який буде чітко та коректно визначати напрямок зору в реальному часі, застосовуючи метод Хафа та допоміжний метод метод Гауса-Габора. Мова Python є досить поширеною для використання при створенні нейронних мереж, що стане в нагоді мені надалі. Також ця мова містить достатню бібліотеку з літературою, яка зможе допомогти мені при розробці програмного продукту обраних методів та скоротить процес ознайомлення із функціоналом.

Основною бібліотекою, яка буде використовуватись у моєму програмному коді – це бібліотека комп'ютерного зору OpenCV. Саме обрана мною мова програмування Python з цієї бібліотекою відкриває широкі можливості для обробки зображень для оптимізації систем.

Окрім вищеперелічених вагомих переваг, також існують деякі недоліки, такі як низька швидкодія, адже мова Python може бути менш ефективною у спритності, особливо при обширних обчисленнях. Та також існує думка, що через те, що основний інтерпретатор Python використовує достатньо велику кількість даних, які

є потіконебезпечними, мова не має властивості багатопоточності. Через це в моменті може виконуватись лише одна з нитей коду, незважаючи на комплектацію та потужності комп’ютера. В даній розробці використовувалась версія Python 3.12.0, реліз якої був у 2023 році.

Також було використано систему керування базами даних MySQL для роботи з базою даних, в якій будуть зберігатися вихідні дані результату роботи програми.

**MySQL** — це система керування базами даних (СКБД), яка використовує мову структурованих запитів (SQL) для управління реляційними базами даних [8].

У базі даних я зберігаю наступні дані: центр координати зіниці, поточне зображення, зображення із визначеним кругом та центром, маска та контур для формування дата-сету, який знадобиться мені для подальшого навчання згорткової нейронної мережі.

### 3.2 Інструменти та середовище розробки

Середовищем розробки мого програмного забезпечення став PyCharm. За допомогою цього зручного та простого у використанні та ознайомленні середовищі було успішно розроблено усі поставлені перед собою задачі, такі як: алгоритм пошуку темних кругів, обрахування центру кругу та його координат.

**PyCharm** – це найпопулярніше та найзручніше середовище для розробки програмного забезпечення на мові Python. Дане середовище містить в собі усі необхідні інструменти для створення надійного та коректно працюючого програмного коду для поставлених задач. PyCharm має кілька варіантів ліцензування та в нашому випадку було використано безкоштовну версію Community Edition з дещо обмеженим набором можливостей, але яких цілком достатньо для даної розробки [9].

Процес розробки програмного коду став більш комфортним та швидким через наступні характеристики:

– зручне та швидке редагування коду: миттєві перевірка та аналіз коду з підсвічуванням;

– інтелектуальна навігація по коду;

– підтримка наукових бібліотек, а саме: SciPy, NumPy, Matplotlib;

– вбудований спеціальний інструмент для зручної роботи із базами даних;

– наявність спеціального відлагоджувача для зручного та швидкого тестування коду;

– крос-платформаність Windows, Mac OS X, Linux.

Саме ці функції значно спрощують процес розробки, зробивши його більш ефективним і зручним та дозволяючи розробникам легко орієнтуватися в складних проектах і знаходити необхідні елементи коду.

### 3.3 Використані бібліотеки

У цій роботі були застосовані важливі бібліотеки для успішної реалізації поставлених завдань. Тому за допомогою наступних бібліотек було розроблено частину розробки із пошуком темних кругів, обрахування центру кругу та його координат. Основні бібліотеки, які використовувались під час розробки програмного коду:

– бібліотека комп’ютерного зору OpenCV;

– бібліотека для машинного навчання NumPy.

Обрані бібліотеки є цілком відомими та ефективними для виконання подібних проєктів та забезпечують коректну роботу програмного продукту.

### 3.3.1 Бібліотеки для обробки зображень

Основою мого програмного забезпечення із алгоритмічним пошуком напрямку зору в реальному часі – це широковикористовувана бібліотека комп’ютерного зору OpenCV.

**OpenCV (Open Source Computer Vision Library)** – це відкрита бібліотека для комп'ютерного зору та машинного навчання, яка містить понад 2500 оптимізованих алгоритмів. Ця бібліотека використовується для швидкої та коректної обробки зображень та відео у реальному часі та розпізнавання об’єктів, тексту та іншої інформації [10].

Вона підтримує безліч програмних платформ, включаючи Windows, Linux, Mac OS, iOS, та Android. Завдяки своїй відкритості та підтримці багатьох мов програмування, OpenCV є популярним засобом розробки. Ця бібліотека комп’ютерного зору була створена компанією Intel та в наш час отримує підтримку з боку розробників Inseez та Willow Garage. Загалом OpenCV написана на C++ та весь її загальний інтерфейс також, але також наразі реалізовано інтерфейс мовами Python, Java і MATLAB / OCTAVE тощо.

Усі присутні оптимізовані алгоритми у бібліотеці OpenCV є як і класичним, так і новітніми та усі вони мають одну основну мету – коректну роботу з комп’ютерним зором та машинним навчанням. Зазвичай ця велика кількість алгоритмів використовується для:

– виявлення і розпізнавання певних об’єктів;

– ідентифікації обличчя, жестів та рухів;

– виявлення руху;

– фільтрації зображень та виявлення країв;

– спостерігання на напрямком зору та руху очей;

– побудови 3D моделей об’єктів;

– аналізу руху;

– відстеженням переміщення камери.

Вищеперелічені приклади використання алгоритмів – це ще далеко не всі можливості, на які здатна дана бібліотека. Та загалом бібліотека комп'ютерного зору призначена у моїй розробці для роботи з зображеннями та розпізнаванням об'єктів на них, зокрема напрямку зору та руху очей.

Найзручнішою мовою для використання цієї бібліотеки є Python, яку і була використана для створення мого програмного забезпечення, хоча бібліотека має інтерфейси і у інших популярних мовах програмування, таких як:

– C++;

– Java;

– MATLAB.

Отже, дана бібліотека є дуже потужною та універсальною та знаходить своє застосування у багатьох сферах, включаючи науку, медицину, безпеку, розваги та робототехніку. Завдяки своїй відкритості та багатофункціональності, OpenCV є незамінним інструментом для багатьох фахівців і дослідників, що підтверджує широке використання та практичність цього засобу в програмуванні.

### 3.3.2 Машинне навчання бібліотека NumPy

Оскільки Python є інтерпретованою мовою, то математичні алгоритми часто працюють в ньому набагато повільніше ніж у компільованих мовах, таких як C або навіть Java. Тому для вирішення цієї проблеми було використано бібліотека Python NumPy, яка забезпечує підтримку багатовимірних масивів і безлічі функцій та операторів для роботи з ними. Таким чином будь-який алгоритм, який може бути виражений в основному як послідовність операцій над масивами і матрицями, працює так само швидко, як еквівалентний код, написаний на C.

Завдяки бібліотеці NumPy було розроблено частину програмного коду, яка відповідала за створення ядра для операції морфологічної обробки, використання

масивів даних, приведення типу та округлення координат кіл, знайдених за допомогою алгоритму Хафа та інші.

**NumPy** – це бібліотека для мови програмування Python, яка додає підтримку великих багатовимірних масивів і матриць, а також широкий набір високорівневих математичних функцій для роботи з ними [11].

NumPy є найпоширенішою бібліотекою для наукових обчислень у мові програмування Python, тому часто використовується в поєднанні з іншими бібліотеками, такими як SciPy, pandas і Matplotlib. Завдяки своїй продуктивності та багатофункціональності, NumPy є невід'ємною частиною екосистеми Python для наукових досліджень і аналізу даних.

### 3.4 Вимоги до програмно-апаратної частини

Фактично основною вимогою для коректної роботи моєї розробки є наявність камери. Тому загальні визначені вимоги до комп’ютера, на якому буде працювати даний програмний продукт, такі як:

– наявність камери;

– операційна система: Windows 7+, Linux, MacOs;

– процесор: Intel core 2 duo 2+;

– HDD or SDD: 10 gb;

– RAM: 2 gb.

Аналізуючи описані вище вимоги, можна зрозуміти, що створене програмне забезпечення є достатньо доступним.

# 4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В даному розділі я розглянула основні програмні частини коду серверної частини розробки та програмний код бази даних. Серед використаних алгоритмів в розділі буде представлено кодування методу Хафа та розглянуто усі параметри, який він потребує. Також представлено допоміжні алгоритми, які деталізують роботу вищезазначеного методу та покращують візуалізацію програмного продукту.

Для розробки програмного коду я розробила блок-схему для визначення послідовності виконання програмних частин алгоритму (рисунок 4.1).

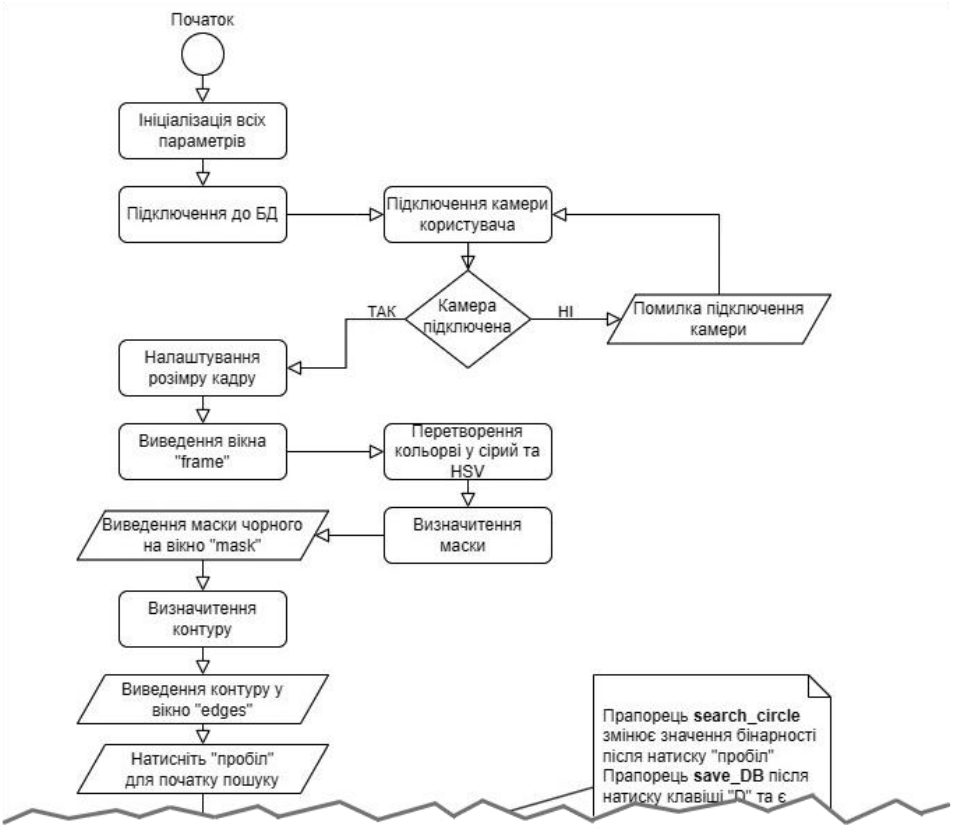



Рисунок 4.1 – Підготовчі алгоритми для пошуку зіниці

Ця частина блок-схеми підводить програмний продукт до основного алгоритму із визначенням напрямку зору. У ній ми підключаємо бібліотеки та БД, ініціалізуємо дані. Наступним кроком є перевірка наявності камери для зчитування зображення користувача. Далі ми визначаємо розмір кадру та переводимо зображення у потрібний для нас формат сірого та HSV. Формуємо маску за діапазоном чорного та робимо контур з маски. Далі очікуємо натиску пробіл для входу до основної частини алгоритму (рисунок 4.2).

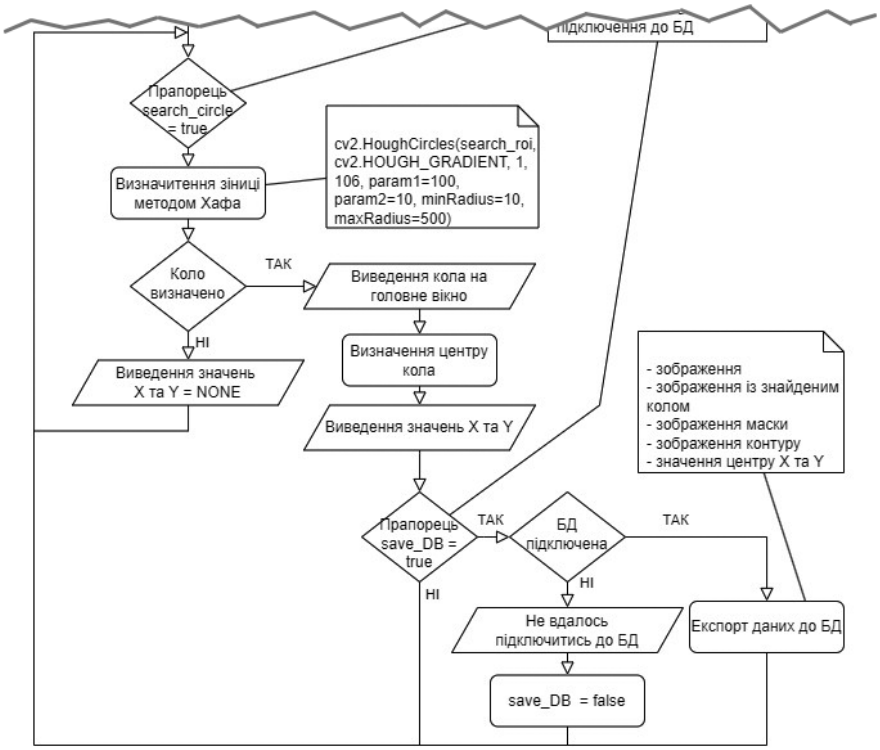



Рисунок 4.2 – Алгоритм пошуку зіниці

Після натиску на клавіші «пробіл» ми входимо в процес пошуку зіниці методом Хафа. Алгоритм забезпечує постійне визначення кола та знаходження його центру, що дає змогу працювати програмному продукту в реальному часі.

На блок-схемі видно частину розробки із експортом успішних даних до бази даних для формування дата-сету. Також у коментарях зазначено усі необхідні деталі для формування програмного коду.

Маючи алгоритм у вигляді блок-схеми я починаю програмувати його. Для початку я підключаю бібліотеки NumPy та бібліотеку комп’ютерного зору OpenCV за допомогою наступних рядків коду: import numpy as np, import cv2. Для підключення доступу до бази даних я імпортую бібліотеку MySQL: import mysql.connector.

Мій програмний продукт зчитує зображення з камери користувача, тобто ми маємо підключити камеру, це я роблю за допомогою бібліотеки cv2: cap = cv2.VideoCapture(0), де cap – об'єкт захоплення відео. Під час програмування основної ділянки коду я перевіряю наявність зв’язку з камерою, або його доступність: if cap.isOpened() is True. Ширину кадру захоплення я задаю наступними ділянками коду: ret = cap.set(3, 640) – для встановлення ширини кадру та ret = cap.set(4, 480) – для його висоти.

Для обробки зображень отриманих з OpenCV, я написала рядок програмного коду, який створює ядро для операцій морфологічної обробки: kernel = np.ones((5, 5), np.uint8).

Ядро програмного коду починається із перевірки прапорця. Це означає, що користувач натисне клавішу пробіл для початку пошуку його зіниці оку. Я реалізувала це для того, щоб програма не навантажувалась до того, як це не захоче користувач. Тобто юзер спочатку підносить голову до камери, перевіряє, чи потрапило його око в контур для пошуку зіниці.

Обмеження по контуру: search_roi = edges[140:340, 170:470] та зображення цього контуру на екрані для координації користувача – cv2.rectangle(frame, (170, 140), (470, 340), (0, 255, 0), 5)). Після того, як користувач підвів пошуковий об’єкт до екрану – він може натиснути клавішу для початку пошуку. Значення є бінарним та приймає початкове значення False.

Фактично відео з камери це є потоком зображень, а отже ми зчитуємо його за наступним кодом: ret, frame = cap.read(). Дане зображення ми переводимо його у сірий та спрощуємо кольори, що дасть нам можливість простіше визначати кольоровий діапазон нашого об'єкту. Переведення у сірий – gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY), та спрощення – hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV).

Наступний етап задати діапазон значень темного кольору для зчитування саме зіниці ока: lower_black = np.array([0, 0, 0]) – для нижньої межі темного кольору та upper_black = np.array([179, 255, 30]) – для верхньої границі. Я створюю маску, яка відображає частини кадру, де кольори відповідають заданим межам чорного кольору та обробляю цей | чорний об'єкт згладжуючи його та видалення шумів: bila = cv2.bilateralFilter(mask, 10, 200, 200) та opening = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel). Цю маску я вивожу на окремий екран для демонстрації знайдених чорних об’єктів: cv2.imshow('mask', mask).

Далі зважаючи на результати знайденої маски я шукаю круги Хафа: Circles = cv2.HoughCircles(search_roi, cv2.HOUGH_GRADIENT, 1, 106, param1=100, param2=10, minRadius=10, maxRadius=500).

Знайдені кола я обмальовую та позначаю його центр. Також я визнаю його радіус. Ці результати я вивожу на екран. Тобто малюється центр кола та червоним кольором виводиться його коло (рисунок 4.1).

```
if circles is not None:
    circles = np.uint16(np.around(circles))
    for circle in circles[0]:
        x = int(circle[0]) + 170  # Перетворення координат кола на відносні до всього кадру
        y = int(circle[1]) + 140  # Перетворення координат кола на відносні до всього кадру
        r = int(circle[2])
        cv2.circle(frame, center: (x, y), radius: 7, color: (0, 0, 255), thickness: 3)
        cv2.circle(frame, center: (x, y), radius: 3, color: (255, 255, 0), -1)
        text = 'x: ' + str(x) + ' y: ' + str(y)
```

Рисунок 4.1 – Визначення кола та його центру

Вхідними параметрами алгоритму є область зображення, Хафа градієнт, масштаб, мінімальна відстань між центрами кіл, порогові параметри для виявлення країв та мінімальні, максимальні значення радіусів.

На екран я вивожу значення координати центу знайденої зіниці, для цього я встановлюю шрифт: font = cv2.FONT_HERSHEY_SIMPLEX, та вивожу на екран це значення: cv2.putText(frame, text, (10, 30), font, 1, (0, 255, 0), 2, cv2.LINE_AA), а у випадку, якщо не знайдено зіниці – cv2.putText(frame, 'x: None y: None', (16, 30), font, 1, (0, 255, 0), 2, cv2.LINE_AA).

Для того щоб перенести поточні дані, які я знаходжу до бази даних для наповнення дата-сету, я натискаю клавішу D. Після цього спрацьовує ділянка коду із вивантаженням наступних даних до бази даних.

```
def save_to_database(image, circle_image, x, y, mask_image, edges_image):
    try:
        connection = mysql.connector.connect(
            host='your_host',
            database='your_database',
            user='your_user',
            password='your_password'
        )

        if connection.is_connected():
            cursor = connection.cursor()
            query = ("INSERT INTO circles_data (image, circle_image, x_coord, y_coord, mask_image, edges_image) "
                     "VALUES (%s, %s, %s, %s, %s, %s)")
            cursor.execute(query, (image, circle_image, x, y, mask_image, edges_image))
            connection.commit()
            print("Data saved to database successfully")
    except Error as e:
        print("Error while connecting to MySQL", e)
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
```

Рисунок 4.2 – Експорт даних до дата-сету

Я передаю наступні параметри для заповнення дата-сету для навчання згорткової нейронної мережі: центр координати зіниці, зображення поточне, зображення поточне із зображення круга та його центру, маску об'єкту та результати алгоритму кругу Хафа.

# 5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

В даному розділі буде представлено пояснення роботи користувача з системою. Я опишу послідовність виконання дій користувача для отримання позитивного результату.

Мій користувацький інтерфейс є легким, комфортним та зрозумілим для будь-якого користувача. Для початку роботи із системою потрібно запустити програмний застосунок. Після запуску користувач отримує три вікна: головний екран із зображенням в реальному часі, екран із маскою та екран з контуром (Рисунок 5.1).

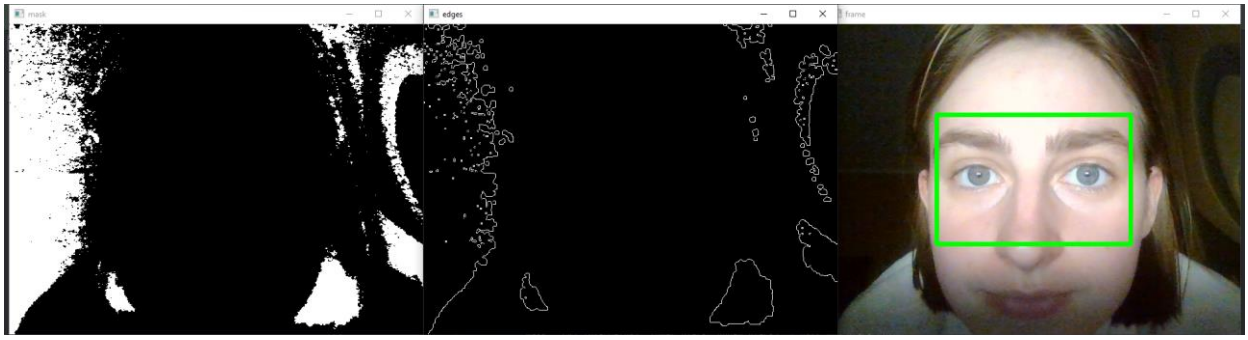



Рисунок 5.1 – Початок роботи програми

На головному вікні користувач бачить обмежувальну зону для пошуку зіниці. Користувач займає зручне положення перед екраном та наближує своє обличчя ближче до камери для того, щоб обмежувальна лінія була близькою до контурів ока. Щоб розпочати процес пошуку зіниці, користувач натискає клавішу «пробіл». На екрані система малює знайдене коло червоним кольором та його центр зеленим.

У верхньому лівому куті головного вікна можна побачити координати центру знайденого кола (Рисунок 5.2).

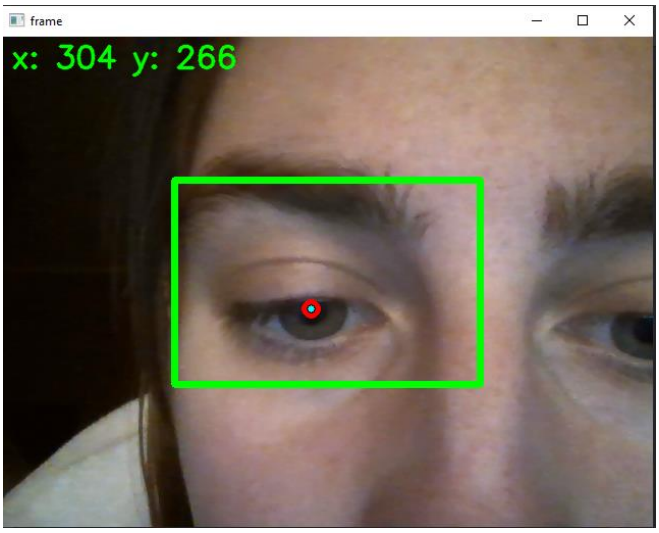



Рисунок 5.2 – Знайдена зіниця ока

При зміні напрямку зору програма малює лінію між попереднім центром зіниці та поточним. Таким чином, користувач може в реальному часі побачити зміну напрямку зору. У випадку, якщо зіницю користувача не розпізнано, то координати X та Y матимуть значення None. Для того, щоб зупинити пошук зіниці, користувач натискає «пробіл» або закриває головне вікно натиском на «хрестик».

Допоміжне вікно «mask» показує маску темних об’єктів. На цьому зображенні помітні зниження шумів та згладжування контурів. На сусідньому вікні «edges» користувач бачить контури об’єктів. Саме ці контури обробляються методом перетворення Хафа для пошуку кругів.

У програмі існує функціонал експортування успішних результатів роботи програмного продукту до бази даних. Таким чином, користувач наповнює дата-сет потрібними даними для подальшого навчання згорткової нейронної мережі. Для запуску процесу вивантаження даних потрібно натиснути клавішу «D».

# ВИСНОВКИ

Під час роботи на дипломним проектом було розроблено програмний продукт із пошуком напрямку зору в реальному часі, використовуючи метод Хафа та метод Гауса-Габора.

1. Аналізуючи поставлену задачу із визначенням напрямку зору в реальному часі, було обрано найоптимальніший метод перетворення Хафа для визначення круга зіниці та її центру. Для кращого та швидшого зчитування зображення для пошуку кіл методом Хафа було застосовано додатковий метод Гауса-Габора, що згладжує та видаляє зайві шуми на зображенні.

2. Після ознайомлення з існуючими програмними засобами, які використовують для подібних розробок, було визначено найефективніші засоби розробки. Інтерпретована мова програмування Python є досить швидкою та зручною для реалізації поставлених цілей. Дана мова містить безліч бібліотек, які зможуть закрити потреби програмного забезпечення. Для реалізації пошуку напрямку зору в реальному часі на основі нейронної мережі було обрано бібліотеку комп'ютерного зору OpenCV, що забезпечує швидке та коректне розпізнавання об'єктів та обробку зображень та відео у реальному часі. Для швидшого обрахунку складних математичних операцій було застосовано бібліотеку Python NumPy.

3. В результаті роботи було створено програмний продукт, який забезпечує чітке та коректне знаходження напрямку зору в реальному часі. Програма виводить 3 вікна з потрібними зображеннями, а саме: маска, контур та зображення об'єкта в реальному часі. Пошук центру зіниці ока починається після того, як користувач натискає клавішу пробіл. Таким чином, програма знаходить круги та їх центри методом перетворення Хафа, що і є центром зіниці ока. Також, окрім визначених кругів та їх центрів, на екрані відображаються координати центру зіниці. Програмний продукт також забезпечує можливість збереження вихідних даних у дата-сеті після натиску клавіші D.

4. Серверно-апаратна частина моєї розробки показала себе ефективною у пошуку напрямку зору в реальному часі. Обрахунки запропонованих мною методів здійснюються швидко та точно, що забезпечує швидке та надійне визначення місцезнаходження зіниці ока.

На мою думку, дана розробка є вагомою та ефективною для подальшого масштабування. Створення окулярів, що будуть визначати напрямок зору людини в реальному часі та направляти в цю точку влаштовані світлові промені, є важливим для покращення якості роботи багатьох сфер діяльності. А особливо така розробка зможе допомогти прифронтовим медикам рятувати життя військовим в польових умовах.

Отже, визначення напрямку зору в реальному часі є актуальною розробкою та знайде своє призначення у багатьох сферах нашого повсякденного життя.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт про реалізацію R-TOD: веб-сайт. URL:https://paperswithcode.com/task/real-time-object-detection (дата звернення 26.05.2024).

2. Сайт про організація часу, діаграма Ганта: веб-сайт. URL:https://blog.tmetric.com/how-time-management-in-software-development-should-actually-work-2/ (дата звернення 27.05.2024).

3. Сайт про розробку діаграм: веб-сайт. URL: https://www.smartdraw.com/ (дата звернення 27.05.2024).

4. Duda, R. O. and P. E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures, California: Comm. ACM, 1972. 5 p.

5. Сайт про Гаусове згладжування: веб-сайт. URL: https://web.archive.org/web/20071211124946/http://www.cee.hw.ac.uk/hipr/html/gsmooth.html (дата звернення 28.05.2024).

6. Lee C. J., Wang S. D. Fingerprint feature extraction using Gabor filters, Netherlands, Rotterdam: Elsevier, 1999. 35(4).

7. Mark Lutz Learning Python 5th Edition. Sebastopol: O'Reilly, 2021. 1542 p.

8. Сайт про програмування: веб-сайт. URL: https://Metanit.com/ (дата звернення 28.05.2024).

9. Середовище розробки PyCharm: веб-сайт. URL: https://www.jetbrains.com/pycharm/ (дата звернення 30.05.2024).

10. Сайт про OpenCV: веб-сайт. URL: https://opencv.org/ (дата звернення 30.05.2024).

11. Бібліотека NumPy: веб-сайт. URL: https://numpy.org/ (дата звернення 30.05.2024).

# ДОДАТОК А

## Реалізація програми для визначення напрямку зору на основі нейронної мережі

Програмні засоби:

– мова програмування – Python;

– середовище розробки – PyCharm;

– бібліотека комп’ютерного зору – OpenCV;

– виконання математичних/логічних операцій з великими масивами даних – Numpy;

– база даних – MySQL.

```
import numpy as np
import cv2
import mysql.connector
from mysql.connector import Error

def save_to_database(image, circle_image, x, y, mask_image,
edges_image):
    try:
        connection = mysql.connector.connect(
            host='your_host',
            database='your_database',
            user='your_user',
            password='your_password'
        )

        if connection.is_connected():
            cursor = connection.cursor()
            query = "INSERT INTO circles_data (image,
circle_image, x_coord, y_coord, mask_image, edges_image)
VALUES (%s, %s, %s, %s, %s, %s)"
            cursor.execute(query, (image, circle_image, x,
y, mask_image, edges_image))
            connection.commit()
            print("Data saved to database successfully")
    except Error as e:
        print("Error while connecting to MySQL", e)
    finally:
```

```
        if connection.is_connected():
            cursor.close()
            connection.close()

cap = cv2.VideoCapture(0)
ret = cap.set(3, 640)
ret = cap.set(4, 480)
font = cv2.FONT_HERSHEY_SIMPLEX
kernel = np.ones((5, 5), np.uint8)
search_circle = False

if cap.isOpened() is True:
    while True:
        ret, frame = cap.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        lower_black = np.array([0, 0, 0])
        upper_black = np.array([179, 255, 30])
        mask = cv2.inRange(hsv, lower_black, upper_black)
        opening = cv2.morphologyEx(mask, cv2.MORPH_OPEN,
kernel)
        bila = cv2.bilateralFilter(mask, 10, 200, 200)
        edges = cv2.Canny(opening, 50, 100)

        search_roi = edges[140:340, 170:470]

        cv2.rectangle(frame, (170, 140), (470, 340), (0,
255, 0), 5)

        if search_circle:
            circles = cv2.HoughCircles(search_roi,
cv2.HOUGH_GRADIENT, 1, 106, param1=100, param2=10,
minRadius=10, maxRadius=500)

            if circles is not None:
                circles = np.uint16(np.around(circles))
                for circle in circles[0]:
                    x = int(circle[0]) + 170
                    y = int(circle[1]) + 140
                    r = int(circle[2])
                    cv2.circle(frame, (x, y), 7, (0, 0,
255), 3)
```

```
                    cv2.circle(frame, (x, y), 3, (255, 255,
0), -1)
                    text = 'x: ' + str(x) + ' y: ' + str(y)
                    cv2.putText(frame, text, (10, 30), font,
1, (0, 255, 0), 2, cv2.LINE_AA)
            else:
                cv2.putText(frame, 'x: None y: None', (16,
30), font, 1, (0, 255, 0), 2, cv2.LINE_AA)

        cv2.imshow('frame', frame)
        cv2.imshow('mask', mask)
        cv2.imshow('edges', edges)
        k = cv2.waitKey(5) & 0xFF
        if k == 27:
            break
        elif k == 32:
            search_circle = not search_circle
        elif k == ord('d') or k == ord('D'):
            if search_circle and circles is not None:
                save_to_database(frame, frame, x, y, mask,
edges)
    cap.release()
    cv2.destroyAllWindows()
else:
    print('cap is not opened!')
```

# ДОДАТОК Б

## Авторське свідоцтво на твір

МІНЕКОНОМІКИ
НАЦІОНАЛЬНИЙ ОРГАН ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ
ДЕРЖАВНА ОРГАНІЗАЦІЯ **«УКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ОФІС ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ ТА ІННОВАЦІЙ»**

(УКРНОІВІ)

Розписка про одержання електронної заявки на реєстрацію авторського договору про передачуправа на використання твору

Вх. № 500767 Дата одержання 11.06.2024 14:34:49

| | |
|---|---|
| **Номер заявки** | **r202400176**<br>*(в подальшому обов'язково посилатися на цей номер)* |
| Відправник | Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського" |
| Назва реєстрацію авторського договору про передачу права на використання твору | Комп’ютерна програма «Визначення напрямку зору на основі нейронної мережі» |