

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Засідувач кафедри

 /В. І. Шинкаренко/

« 18 » 12 20 20 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань 12 Інформаційні технології

Спеціальність 121 Інженерія програмного забезпечення

Тема Розробка методів відстеження відповідності фокусу зору зі структурою
інформації на моніторі

Theme Methods development of gaze point monitoring in conformity with the
information content on the PC screen

Керівник дипломної роботи

проф.  В. І. Шинкаренко

Нормоконтролер

доц.  О. С. Куроп'ятник

Студентка групи ПЗ1921

 Ю. О. Снігур

Student

Snihur Yuliia

Дніпро – 2020

Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Факультет Технічна кібернетика кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

 проф. Шинкаренко В.І.

(підпис)

«10» новта 2020 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС Магістр
(освітньо-кваліфікаційний рівень)

студентки групи ПЗ1921 Снігур Юлії Олександрівни
(номер групи) (ПІБ)

1 Тема дипломної роботи: Розробка засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК

затверджена наказом по університету від «10» жовтня 2020 р. № 779ст.




2 Термін подання студентом закінченої роботи 16 грудня 2020р.

3 Вихідні дані до дипломної роботи

4 Зміст пояснювальної записки (перелік питань до розробки) Актуальність, постановка задачі, аналіз сучасних аналогів, обґрунтування обраного методу, розробка інструментальних засобів, дослідження та аналіз результатів, охорона праці та безпека в надзвичайних ситуаціях, висновки

5 Перелік демонстраційного матеріалу презентація на тему «Розробка засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК», відео з прикладом роботи розробленої системи пошуку відповідності фокусу зору зі структурою інформації на моніторі

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Визначення витрат на проектування програмки	доцент Гісенний М.В.	20.10.20 	02.12.20 
Охорона праці та безпека в надзвичайних ситуаціях	старший викладач Музикін М.І.	19.10.20 	03.11.20 

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва розділів дипломного проекту	Термін виконання розділів проекту (роботи)	Примітка
1	Вступ	01.11.2019 – 15.11.2019	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	20.11.2019 – 15.12.2019	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	16.12.2019 – 30.12.2019	
4	Постановка задачі, технічне завдання	03.01.2020 – 15.01.2020	30%
5	Техніко-економічні показники	15.01.2020 – 24.01.2020	
6	Розробка інструментальних засобів дослідження	02.02.2020 – 03.07.2020	
7	Виконання досліджень	08.07.2020 – 16.08.2020	60%
8	Оформлення тез доповідей	19.08.2020 – 23.08.2020	
9	Оформлення статті у фаховий журнал	23.08.2020 – 30.08.2020	
10	Оформлення пояснювальної записки	01.09.2020 – 01.12.2020	
11	Розробка демонстраційних матеріалів	14.12.2020 – 18.12.2020	100%

Дата видачі завдання «10» новемб. 2019 р.

Керівник дипломного проекту


(підпис)

В.І. Шинкаренко
(підп.)

Завдання прийняв до виконання


(підпис)

Ю.О. Снігур
(підп.)

РЕФЕРАТ

Об'єкт дослідження – процес контролю за навчанням студентів.

Предметом дослідження є засоби реалізації автоматизованої системи відстеження відповідності фокусу зору зі структурою інформації на моніторі.

Метою роботи є дослідження засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі з використанням апаратних засобів, які доступні кожному для покращення засвоювання інформації студентами під час навчання.

Методи дослідження: обробка відеокадрів, розпізнавання зіниці ока на відеокадрі, підрахунок координат фокусу зору та відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера.

Результати та їх новизна: дослідження робить внесок у підвищення точності відстеження фокусу зору. Результати дослідження дозволяють зробити висновки щодо необхідності редагування інформаційного контенту, для покращення концентрації уваги студента під час дистанційного навчання.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та 5 додатків:

- у вступі описується сутність розробки, її актуальність (2 с.);
- у першому розділі висвітлено аналіз сучасного стану дослідження проблеми (16 с.);
- у другому розділі надано обґрунтування методу відстеження відповідності фокусу зору зі структурою інформації на моніторі (18 с.);
- у третьому розділі представлено проектування й розробка інструментального забезпечення для дослідження (18 с.);
- у четвертому розділі описано виконані дослідження (17 с.);
- у п'ятому розділі розкриті питання охорони та безпеки праці (8 с.);
- додатки містять технічне завдання, робочий проект, тези двох докладів та статті, підготовлені до друку.

Таблиць – 16, рисунків – 54, бібліографія – 42 джерела.

Ключові слова: відстеження, фокус зору, зіниця ока, калібрування, відеокадр.

ЗМІСТ

Вступ.....	10
1 Аналіз сучасного стану дослідження окулографії.....	12
1.1 Основні відомості про око, та його рух	12
1.2 Історія методів дослідження окулографії (айтрекінгу)	14
1.3 Методи реєстрації переміщення ока	16
1.4 Ринок сучасних айтрекерів.....	19
1.5 Сфери застосування	22
1.6 Окулографія у сфері навчання	23
1.7 Сучасні аналоги та проблеми використання окулографії у сфері навчання....	23
1.8 Огляд бібліотек комп'ютерного зору для дослідження розробки засобів відстеження фокусу зору	24
Висновки до першого розділу.....	27
2 Обґрунтування методу відстеження відповідності фокусу зору зі структурою інформації на моніторі.....	29
2.1 Модель відстеження фокусу зору людини	29
2.2 Розпізнавання рис обличчя.....	30
2.2.1 Процес розпізнавання рис обличчя	31
2.3 Розпізнавання очей.....	34
2.3.1 Знаходження координат центра ока	34
2.3.2 Знаходження координат центру зіниці ока	35
2.4 Процес калібрування.....	38
2.5 Підрахунок координат фокусу зору	41

	6
2.6 Оцінка похибки знаходження координат фокусу зору	43
2.7 Пошук відповідності фокусу зору зі структурою інформації на моніторі.....	44
2.8 Системні обмеження для відстеження фокусу зору	46
2.9 Опис апаратних та програмних засобів	46
Висновки до другого розділу	47
3 Розробка інструментальних засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі	48
3.1 Зовнішнє проектування	48
3.1.1 Формалізація задачі.....	48
3.2 Базова архітектура системи.....	51
3.3 Внутрішнє проектування.....	52
3.3.1 Вибір інструментальних засобів розробки	52
3.3.2 Ієрархія та взаємодія модулів системи	53
3.3.3 Використані принципи проектування.....	57
3.4 Розробка інтерфейсу користувача	58
Висновки до третього розділу.....	64
4 Дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі.....	66
4.1 Експеримент – дослідження відстеження фокусу зору користувача з використанням однієї камери при утриманні голови в статичному стані	66
4.2 Експеримент – дослідження відстеження фокусу зору користувача з використанням двох камер при утриманні голови в статичному стані.....	70
4.3 Експеримент – дослідження відстеження фокусу зору користувача з використанням зміни позиції калібрувальної точки по обом осям	71

4.4 Експеримент – дослідження відстеження фокусу зору користувача з використанням збільшеної кількості позицій калібрувальної точки.....	77
4.5 Експеримент – дослідження відстеження фокусу зору користувача з утриманням голови користувача в динамічному стані.	78
4.6 Експеримент – дослідження відстеження фокусу зору користувача з утриманням голови користувача в динамічному стані та зміною розташування додаткової зовнішньої відеокамери	80
Висновки до четвертого розділу.....	81
5 Охорона праці та безпека в надзвичайних ситуаціях.....	83
5.1 Вимоги безпеки при виконанні робіт на робочому місці	83
5.2 Шкідливі виробничі фактори на робочому місці	85
5.2.1 Організація робочого місця.....	85
5.2.2 Вимоги до освітлення у приміщенні	86
5.2.3 Вимоги до мікроклімату приміщення.....	86
5.2.4 Вимоги до шуму та вібрації у приміщенні	87
5.3 Дії працівників в надзвичайних ситуаціях	89
Висновки до п'ятого розділу.....	90
Загальні висновки.....	92
Список використаних джерел	93
Джерела.....	99
Технічне завдання.....	
Робочий проект.....	
Специфікація.....	
Опис програми.....	
Керівництво користувача. Керівництво відстеження фокусу зору користувача....	

Текст програми.....	
Тезиси на XIII Міжнародну науково-практичну конференцію «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті»	
Тезиси на XIV Міжнародну науково-практичну конференцію «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті»	
Стаття підготовлена до друку	

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПК – персональний комп'ютер

ПЗ – програмне забезпечення

P1 – координати позиції калібрувальної точки

P2 – координати фокусу зору

ВСТУП

На сьогоднішній день, у зв'язку з поширенням епідемії коронавірусу Covid-19, закриваються навчальні заклади по всьому світу. В цей час надзвичайно зростає потреба у дистанційному навчанні. Студентам важко концентрувати увагу, коли вони засвоюють матеріал самостійно. Тому актуальним є завдання покращення концентрації уваги студента під час дистанційного навчання. Для вирішення цього завдання дослідники проводять дослідження, щоб відстежити на що студент звертає увагу, що залишається поза увагою, скільки часу він витрачає на засвоєння того чи іншого матеріалу. Для проведення таких досліджень необхідні спеціальні засоби відстеження відповідності фокусу зору зі структурою інформації на моніторі.

Об'єкт дослідження – процес контролю за навчанням студентів.

Предметом дослідження є засоби реалізації автоматизованої системи відстеження відповідності фокусу зору зі структурою інформації на моніторі, які використовують набір методів комп'ютерного зору.

Метою роботи є дослідження засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі з використанням апаратних засобів, які доступні кожному для покращення засвоєння інформації студентами під час дистанційного навчання.

Основним завданням є дослідження та реалізація засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера з використанням доступних апаратних засобів.

Методи дослідження:

- методи обробки відеокадрів;
- методи розпізнавання рис обличчя на відеокадрі;
- методи розпізнавання зіниці ока;
- методи підвищення точності розпізнавання фокусу зору;
- методи підрахунку координат фокусу зору;

- методи відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера.

Наукова новизна.

Запропоновано вдосконалену систему відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера з використанням доступних апаратних засобів: звичайної відеокамери, вбудованої в ноутбук та зовнішньої веб-камери. Запропонована система забезпечує більшу точність відстеження фокусу зору порівняно з іншими системами.

Практичне значення одержаних результатів.

Розроблена система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера дозволяє зробити висновки щодо необхідності редагування інформаційного контенту, для покращення концентрації уваги студента під час дистанційного навчання.

Апробація результатів дослідження та публікації.

Основні положення магістерської роботи доповідалися та були схвалені на засіданнях кафедри КІТ ДНУЗТ (28.11.2019 р., 13.02.20 р., 07.10.2020 р.).

За результатами роботи опубліковано наукові праці – тези доповідей на наукових конференціях [1], [2].

1 АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕННЯ ОКУЛОГРАФІЇ

1.1 Основні відомості про око, та його рух

Орган зору є найважливішим органом чуття людини, адже 80% інформації про навколишній світ людина отримує через зоровий аналізатор.

Око – складний оптичний прилад, основною задачею якого є передавати зображення зоровому нерву. Будова ока зображена на рис. 1.1.

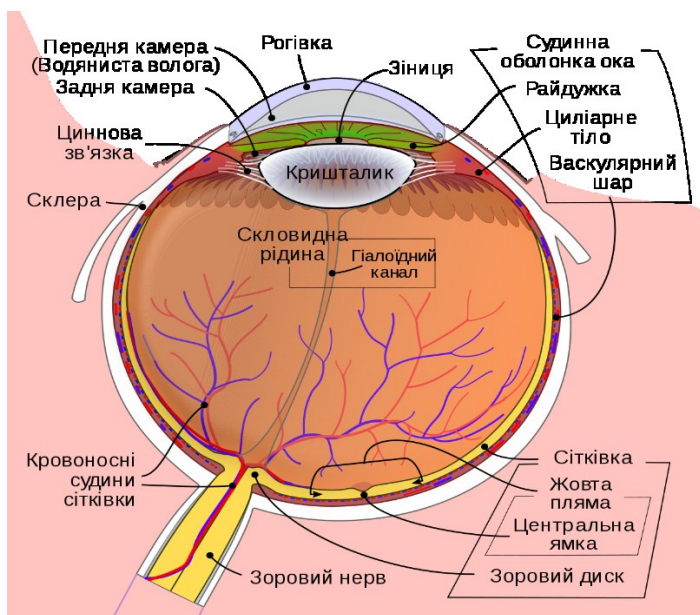


Рисунок 1.1 – Будова ока людини

Основні функції ока:

- оптична система, що проектує зображення;
- система, яка сприймає і «кодує» отриману інформацію для головного мозку;
- «обслуговуюча» система життєзабезпечення.

Будова ока:

- рогівка – прозора оболонка, що покриває передню частину ока;
- передня камера ока – простір між рогівкою та райдужкою;
- райдужка – за формою схожа на коло з отвором всередині – зіницею. Райдужка складається з м'язів, при скороченні і розслабленні яких розмір зіниці змінюється. Райдужка відповідає за колір очей;

- зіниця – отвір в райдужці. Її розміри зазвичай залежать від рівня освітленості, чим більше світла, тим менше зіниця;
- кришталик – «природна лінза» ока. Він прозорий, еластичний, може змінювати свою форму, майже миттєво «наводячи фокус», за рахунок чого людина бачить добре і зблизька, і здалека;
- склоподібне тіло – гелеобразна прозора субстанція, розташована в задньому відділі ока. Склоподібне тіло підтримує форму очного яблука;
- сітківка – складається з фоторецепторів (вони чутливі до світла) і нервових клітин. Клітини-рецептори розташовані в сітківці, в яких відбувається перетворення енергії світла в електричну енергію нервової тканини, тобто відбувається фотохімічна реакція;
- склера – непрозора зовнішня оболонка очного яблука;
- зоровий нерв – за допомогою зорового нерва сигнали від нервових закінчень передаються в головний мозок [3].

Розрізняють декілька видів руху очей:

- фіксації – очі перестають сканувати візуальний простір і статично утримують фовеальний зір, щоб візуальна система змогла отримати детальну інформацію про предмет сприйняття. Фіксації походять від точок погляду. Точка погляду – короточасні просторові ділянки на візуальній осі. Вони мають координати як по осі абсцис, так і по осі ординат та тимчасову позначку. Фіксації утворюються з безлічі точок погляду, мають просторове положення та тривалість – початкову та кінцеву тимчасову позначку;
- саккади – це швидкі рухи очей, які дозволяють сканувати візуальну сцену. Око фокусується на об'єкті тільки на коротку мить, перш ніж швидко перейти до наступного об'єкту. Саккади відбуваються як під час швидкого сну, так і під час читання [4].

Отже, завдяки органу зору можна дослідити чимало інформації та зробити висновки про те, як сприймається інформація про навколишній світ людиною. Процес дослідження органу зору називається окулографія.

1.2 Історія методів дослідження окулографії (айтрекінгу)

Окулографія (айтрекінг) – процес визначення точки, на яку спрямовується погляд чи руху ока відносно голови. Не дивлячись на те, що дана технологія почала застосовуватися останні 15-20 років, перші спроби її застосування були ще в XIX ст. Це було пов'язано з проблемою дослідження диклексії – нездатність оволодіти навичками читання текстів.

В XIX ст. всі дослідження в сфері окулографії проводилися виключно методом спостереження. В 1879 році Луї Еміль Жаваль дослідив, що під час читання надрукованого тексту очне яблуко не рухається монотонно, як вважали раніше. Замість цього, вони роблять короткі зупинки, які пізніше Жаваль назвав фіксаціями, а різкі пересування – саккадами [5]. Дане відкриття призвело до нових питань для дослідження: На яких словах затримується погляд людини?, Як довго це триває?, Навіщо людина повертається до слів, які вона раніше вже читала?

Приклад траєкторії переміщення очного яблука під час читання надрукованого тексту зображено на рис. 1.2.

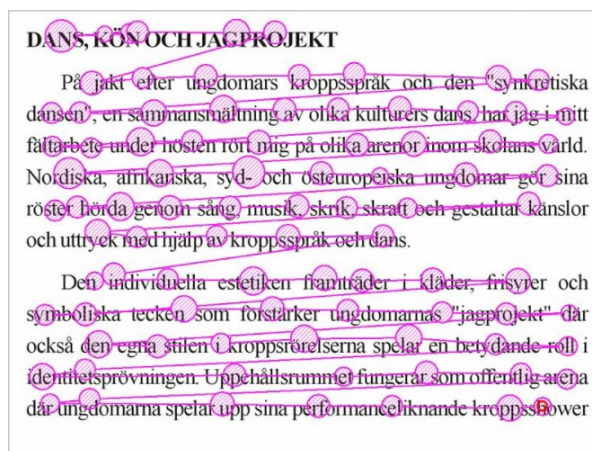


Рисунок 1.2 – Приклад траєкторії переміщення очного яблука під час читання надрукованого тексту

Перший пристрій для відстеження положення зору або очей – айтрекер, був використаний Едмундом Х'ю. Пристрій був схожий на контактні лінзи з алюмінієвим вказівником. Таким чином, Х'ю міг бачити напрямок зору людини під час читання [6]. Принцип роботи пристрою зображений на рис. 1.3.

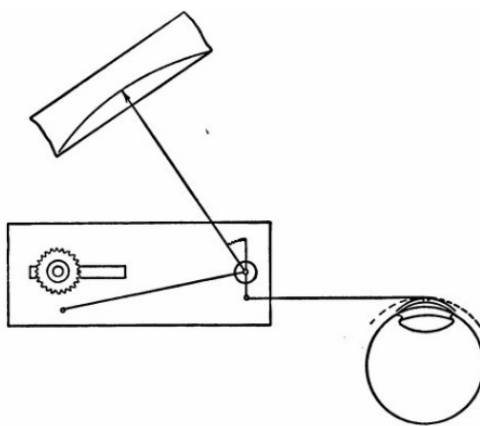


Рисунок 1.3 – Принцип роботи пристрою Едмунди Х'ю

Основним недоліком цього пристрою було те, що він був інвазивним та завдавав незручності для людини.

Після цього немало вчених намагалися розробити неінвазивний пристрій – айтрекер для дослідження процесу читання.

Значний внесок у окулографію зробив Альфред Ярбус. Він розробив свій власний пристрій айтрекер. У цьому пристрої на зіниці закріплювалось маленьке дзеркальце, відблиск якого попадав на фотопапір [7].

Також, Альфред Ярбус сформував гіпотезу про взаємодію візуальної системи та свідомості людини. Вчений довів, що задача, поставлена людині, має величезний вплив на результат експерименту по відстеженню зору. Альфред Ярбус провів чимало експериментів, які довели, що процес переміщення очей відображає процес мислення людини.

Метою експерименту було дослідити об'єкт для вирішення певної задачі. Кожен експеримент тривав 3 хвилини. Об'єкт дослідження для кожного експерименту був однаковий.

З 1980-х років технологія окулографії починає активно розвиватися та прогресувати. З'являються відеоокулографи, які використовують камеру для відстеження положення зіниці.

На рис. 1.4. зображено об'єкт дослідження та результати експериментів для різних поставлених задач.

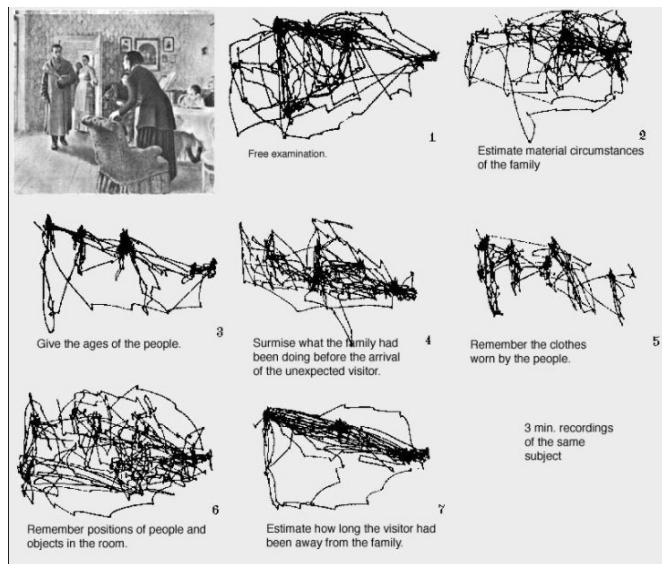


Рисунок 1.4 – Дослідження Ярбуса, які демонструють, що процес переміщення очей відображає процес мислення людини

Отже, процес дослідження погляду людини став розвиватися ще в ХІХ ст. Перші прилади айтрекери для дослідження були інвазивні, що завдавали багато незручностей під час їх використання. Також були спроби створити неінвазивні прилади, більш зручні. Тому у світі з'явилося декілька класифікацій приладів реєстрації переміщення погляду.

1.3 Методи реєстрації переміщення ока

На сьогоднішній день, відеоокулографи – широко використовуються для відстеження зіниці завдяки відеозапису переміщення очей. Камери знімають один, або два ока та реєструють їх переміщення, поки людина спостерігає за об'єктом. Більшість сучасних відеоокулографів використовують контраст між зіницею та радужною оболонкою, який виникає під час інфрачервоного підсвічування. Крім цього, аналізуючи

положення відблиску інфрачервоного підсвічування можна визначити орієнтацію оптичної осі очного яблука.

Окулографічні системи класифікують на:

- системи, які засновані на методі яскравої зіниці;
- системи, які засновані на методі темної зіниці.

Різниця цих систем полягає у розташуванні джерела підсвічування відносно відеокамери. Якщо джерело світла розташоване паралельно оптичній осі камери, око працює як вторинний відбивач світла, який прямує від основного джерела світла та відбивається від сітківки ока, створюючи ефект яскравої зіниці. Якщо джерело світла зсунути відносно оптичній осі камери, зіниця стає чорною, тому що вторинне відбиття світла від сітчатки не потрапляє в камеру. На рис. 1.5 зображено методи темної та світлої зіниці.

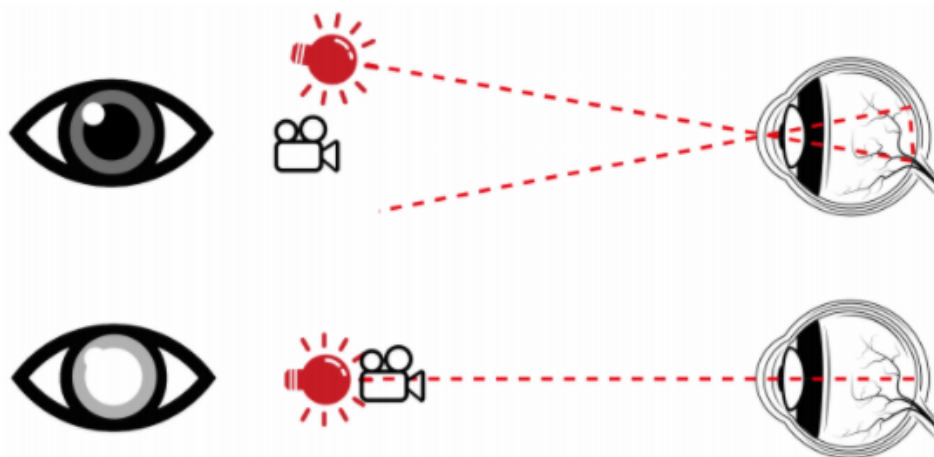


Рисунок 1.5 – Зображення методів темної та світлої зіниці

Системи яскравої зіниці дозволяють відстежувати зір незалежно від кольору райдужної оболонки ока. Ця система також дозволяє відстежувати зір в умовах від повної темноти до високої освітленості.

Прилади для відстеження зору за своєю апаратурою дуже відрізняються. На сьогоднішній день виділяють декілька категорій айтрекерів:

- використання механічного контакту з оком. Прикладом цієї категорії можуть бути контактні лінзи з будованими дзеркалами. Дані прилади використовуються для дослідження динаміки та фізіології переміщення ока;
- використання безконтактних оптичних методів реєстрації переміщення зору. В цій категорії використовується інфрачервоне світло, яке відбивається від очного яблука та реєструється відеокамерою або іншим оптичним сенсором. Прикладом цієї категорії можуть бути окуляри, відеокамери з інфрачервоним світлом. Дана категорія приладів використовується для вирішення задач гейзтрекінгу (знаходження точки перетину оптичної осі очного яблука та площини екрану, на якому розташований об'єкт, за яким спостерігає людина). Перевагою даного методу є висока роздільна здатність;
- використання електричних потенціалів, які обчислюються за допомогою електродів, розташованих навколо очей. Око – це джерело постійного електричного поля. Око можна прирівняти до диполу: позитивний полюс знаходиться на зіниці, негативний – на сітківці. Електричний сигнал можна отримати шляхом використання двох пар електродів, встановлених на шкірі навколо одного ока (рис. 1.6). Даний метод називається електроокулограмою (ЕОГ) [8]. Якщо око рухається з центральної позиції до периферії, то сітківка наближається до одного електрода, а зіниця до іншого. Цей процес змінює орієнтацію диполя, як наслідок, змінюється електричне поле та в результаті змінюється сигнал ЕОГ. Таким чином, аналіз зміни електричного сигналу можна використовувати для аналізу відстеження зору. Даний метод дозволяє досліджувати переміщення очей, навіть коли вони закриті, тому ЕОГ може застосовуватися для дослідження процесу сну людини. Перевага методу – низька вартість обладнання, реєстрація не порушує звичайних умов зорової активності та не залежить від рівня освітленості приміщення. Недоліком даного методу є невисока роздільна здатність.

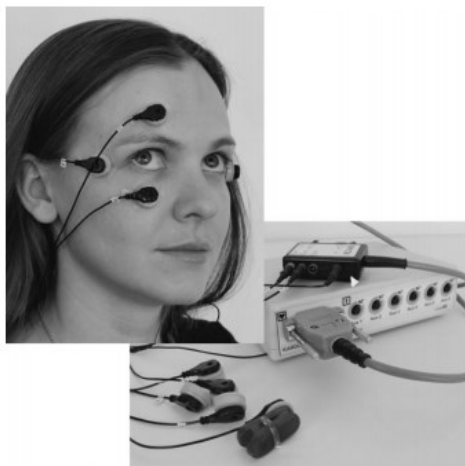


Рисунок 1.6 – Встановлення електродів для реєстрації переміщення очей

1.4 Огляд сучасних айтрекерів

Окулографія – система реєстрації руху очей. Ця система буває двох категорій.

Перша категорія – системи, які необхідно носити – окуляри, або спеціальні оптичні системи. На голові людини монтується система – відеокамера, яка фіксує стан перед людиною та джерела інфрачервоного випромінювання, через яке оптичні оптоволокна підводяться до очей людини, а відбитий від них відблиск «заміщується» в сигнал від відеокамери. Даний вид айтрекерів широко використовується для аналізу складних видів професійної діяльності – під час водіння літаків, автомобілей, управління технічними об'єктами, тощо.

На сьогоднішній день лідером розробки таких систем є шведська компанія Tobii. Компанія займається розробкою продуктів для відстеження напрямку погляду людини за допомогою очей. Прикладом такої продукції є модель Mobile eye tracking Tobii Glasses. (рис.1.7). Точність вимірювань даної моделі досить висока – 0.5° , а частота сканування – 30Гц достатня для використання під час тестування. Ціна даного продукту складає від 20 тис. доларів.



А



Б

Рисунок 1.7 – Айтрекер Mobile eye tracking Tobii Glasses: А – зовнішній вид, Б – вид на голові людини

Основним недоліком приладів, які необхідно носити вважається неможливість прив'язати переміщення очей до екрану монітора за допомогою програмних засобів айтрекерів, тому що голова людини може бути направлена не на монітор.

До другої категорії можна віднести дистанційні айтрекери. Ззовні вони нагадують окремий блок, який розташовується перед людиною на столі під екраном монітору. Прилад айтрекер складається з двох камер з джерелом інфрачервоного світла яке фіксує світлові відблиски, відбиті від очей людини. Схема роботи дистанційного айтрекера зображена на рис. 1.8.



Рисунок 1.8 – Схема роботи дистанційного айтрекера

Існує інший вид дистанційного айтрекера, який складається з декількох окремих відеокамер – від 2-х до 8-ми камер з інфрачервоним світлом. Камери розташовані перед людиною так, щоб максимально підвищити можливість реєстрації переміщення погляду [9]. Дану категорію айтрекерів широко використовують для дослідження двовірних структур.

Порівнюючи з першою категорією айтрекерів, для дистанційних айтрекерів необхідно враховувати параметр допустимий діапазон руху голови, який показує на яку відстань можна відхилити голову вгору-вниз та вліво-вправо.

Прикладом дистанційного продукту є Tobii X2 (рис.1.9). Точність вимірювань даної моделі складає – 0.4° , частота сканування – 30 Гц, допустима відстань до об'єкта – 40-90 см, допустимий діапазон відхилення голови 50x36 см. Ціна даного продукту складає від 10 тис. доларів.



А



Б

Рисунок 1.9 – Айтрекер Tobii X2: А – вмонтований у монітор та ноутбук, Б – у вигляді окремого блоку

Вважають, що дистанційні айтрекери мають більш високий рівень точності ніж окуляри для відстеження зору, але це не так. Точність залежить від типу дослідження, для якого використовують айтрекер. Окуляри для відстеження зору, які зафіксовані на голові добре підходять для дослідження в якому людина може вільно переміщуватися. Дистанційні айтрекери мають обмежений просторовий діапазон, в якому голова респондента може рухатися, при цьому не зменшуються точність отриманих даних.

Таким чином, при виборі приладу айтрекеру необхідно враховувати умови для мети дослідження, тому що від цього залежать вихідні результати. Наприклад, при дослідженні зображень або тексту, представлених на екрані монітора комп'ютера, доцільно використовувати дистанційний айтрекер. Якщо для дослідження респонденту необхідно дивитися на об'єкти навколишнього середовища – доцільно використовувати айтрекер – окуляри.

1.5 Сфери застосування

Основну частину інформації про навколишнє середовище людина сприймає за допомогою очей. Елементи зорової системи – око, нерв та зоровий аналізатор головного мозку – тісно зв'язані між собою, тому дослідження траєкторії руху очного яблука дозволяють зробити висновки про процес розпізнання зорових образів та психічного процесу людини в цілому. Відстеження напряму погляду також дозволяє будувати нові інтерфейси взаємодії між людиною та технічними засобами. У зв'язку з цим, розвиток окулографії є актуальною науково-технічною проблемою.

Система окулографії використовується в наступних сферах:

- для когнітивних досліджень. Окулографія дозволяє зрозуміти яким чином респондент досліджує візуальну інформацію, як змінюються параметри переміщення погляду в залежності від задачі або від психологічних рис;
- у медицині. Окулографія дозволяє створити комунікаційні системи з хворими, які повністю або частково втратили моторну функцію. Також окулографія використовується у медичних дослідженнях для діагностики посттравматичного стресового розладу;
- для досліджень сприйняття UI. Окулографія дозволяє дослідити, які UI елементи потрапляють в зону сприйняття користувача, а що повністю ігнорується, куди користувач дивиться в першу чергу, які елементи заважають, тощо. Результати досліджень допомагають покращувати та оптимізувати макет та візуальний дизайн UI;
- для ергономічних досліджень. Окулографія дозволяє дослідити на які факти звертає увагу водій під час водіння автомобіля, які чинники можуть заважати під час водіння, як перевищення швидкості знижує візуальну увагу. Результати досліджень дозволяють покращити усвідомлення про небезпечні ситуації та безпеку водіїв у майбутньому;

- для досліджень у сфері маркетингу. Окулографія дозволяє досліджувати широке уявлення про звички людей. Ця інформація пов'язана з тим, що повинно приваблювати візуальну увагу у різних ситуаціях [10].

1.6 Окулографія у сфері навчання

В останні роки, технологія відстеження руху очей стала використовуватися у навчанні, задля розуміння стратегії отримання нових знань у різноманітних умовах: від традиційного навчання до цифрового формату. Відстеження руху очей є ефективним інструментом для дослідження зорової уваги та соціальної взаємодії в різноманітних сферах навчання. Використовуючи цей метод, дослідники можуть розуміти, як різні аспекти можуть впливати на результат поглиблення інформації і як покращити процес навчання [11].

Відстеження руху очей дозволяє виявити:

- відмінності у зібранні інформації;
- відмінності у навиках вирішення питань;
- відмінності у стратегії навчання;
- моделі соціальної взаємодії від вчителя до учня;
- як працюють різні навчальні матеріали;
- які елементи приваблюють та утримують увагу.

Технологія відстеження руху очей дозволяє аналізувати, як учені обробляють отриманий матеріал та куди направлена увага під час уроків. Наприклад, надмірна кількість часу, яку учень витратив на обробку певного об'єму інформації може свідчити про те, що учень не розуміє або йому важко засвоїти даний матеріал [12]. Дану інформацію можна використовувати для налаштування навчального контенту для можливості адаптувати, налаштовувати та оптимізувати процес навчання.

1.7 Сучасні аналоги та проблеми використання окулографії у сфері навчання

У сучасному світі для вдосконалення навчального процесу використовують метод окулографії. Завдяки приладу айтрекер можна відстежити увагу студентів та їх

інтереси до різних сфер [13]. Аналізуючи отриманий результат, можна посилити навчальні схеми та покращити навчання.

Як зазначалось раніше, лідером розробки апаратного та програмного забезпечення для застосування методу окулографії є шведська компанія Tobii. Одним із напрямів розробки компанії Tobii Pro є покращення сфери освіти. Tobii Pro має гнучкі рішення, які дозволяють проводити дослідження як у реальному середовищі, так і в лабораторних умовах.

Університет штату Кеннесо використовує окуляри Tobii Pro для вирішення наступних питань:

- що спостерігають студенти під час лекції;
- скільки часу приділяється матеріалу, представленому на уроці;
- як викладач в змозі утримати на собі увагу [14].

Перевагою використання даного приладу є те, що можна відстежити, що насправді приваблює погляд людини. Тому що отримані результати і те, що говорить людина можуть різнитися [15].

Основним недоліком є вартість даного приладу, тому що не кожний навчальний заклад в змозі придбати прилад для кожного студента.

1.8 Огляд бібліотек комп'ютерного зору для дослідження розробки засобів відстеження фокусу зору

Комп'ютерний зір являє собою сукупність програмно-технічних засобів, які забезпечують зчитування зображень в цифровому вигляді, їх обробку і видачу результатів в формі, придатній для практичного використання в режимі реального часу.

Для розробки систем комп'ютерного зору використовують спеціалізовані бібліотеки. Для виявлення найбільш придатної бібліотеки для вирішення поставлених завдань було проведено їх аналіз.

OpenCV – бібліотека алгоритмів комп'ютерного зору, обробки зображень і численних алгоритмів загального призначення з відкритим кодом. OpenCV є однією з найбільш широко відомих бібліотек для вирішення завдань комп'ютерного зору.

Реалізована на мовах програмування C / C++, розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Підтримується різними платформами: Microsoft Windows, Linux, Mac OS, Android, iOS. Має онлайн документацію. Бібліотека включає в себе більше 500 різних функцій і алгоритмів. Зокрема, є безліч оптимізованих алгоритмів, пов'язаних з обробкою і аналізом відеокadrів. Реалізовані функції фільтрації, пошуку контурів, виконання геометричних перетворень, алгоритми аналізу рухів, виявлення об'єктів, спостереження за ними і багато інших. Підтримується робота з xml-файлами [16].

Незважаючи на всі можливості, переваги і плюси бібліотеки OpenCV, слід зазначити її недоліки. Бібліотека OpenCV перевантажена другорядними, додатковими функціями, що необґрунтовано ускладнює її використання та є досить складною для вивчення.

VXL – набір бібліотек, можливості яких використовуються в комп'ютерному зорі і для наукових досліджень. Написані на мові програмування C ++ і можуть бути використані на багатьох операційних системах. VXL забезпечує високу функціональність при обробці матриць. Також VXL працює з зображеннями високої роздільності. Однак необхідно звернути увагу на труднощі в процесі його складання і установки.

LTI-lib – об'єктно-орієнтована бібліотека алгоритмів і структур даних, яка використовується при обробці зображень в сфері комп'ютерного зору [17].

Основною метою розробки даної бібліотеки було створення об'єктно-орієнтованих алгоритмів на мові C++, що в більшості випадків спрощує використання коду і його обслуговування.

Таким чином, аналізуючи бібліотеки комп'ютерного зору, треба звернути увагу на їх швидкодію, оскільки це впливає на якість та швидкість роботи усієї підсистеми розпізнавання об'єктів в цілому. Було проведено порівняльний аналіз роботи OpenCV, OpenCV з використанням компоненту IPP, VXL та LTI для розрахунку швидкості виконання різних операцій:

- дискретне перетворення Фур'є;
- зміна розміру зображення;
- оптичний потік;
- нейронні мережі.

На рис. 1.10 зображено діаграму порівняння швидкодії бібліотек при виконанні різних операцій. Для кожної операції наведено по чотири різних показники ефективності: чотири стовпці для кожного еталону вказують бали – пропорційний часу роботи кожної з даних бібліотек. В усіх випадках OpenCV перевершує інші бібліотеки.

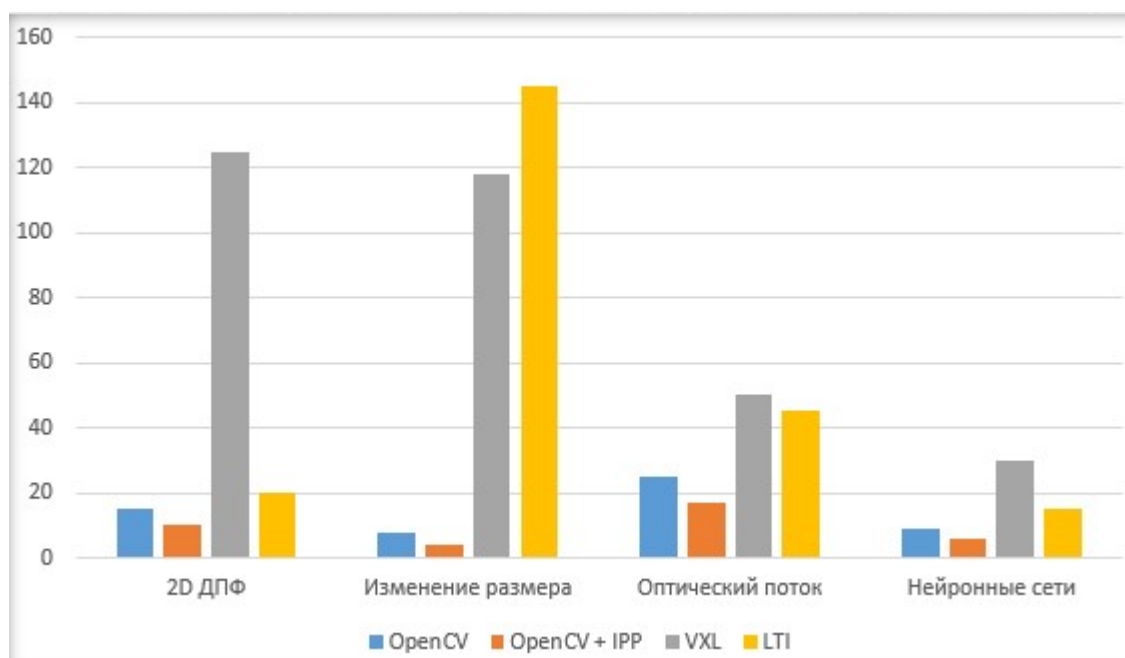


Рисунок 1.10 – Порівняння швидкодії бібліотек при виконанні різних операцій

Отже, аналізуючи бібліотеки комп'ютерного зору, можна зробити висновки, що бібліотека комп'ютерного зору OpenCV є найбільш придатним інструментом для дослідження розробки засобів відстеження фокусу зору.

Згідно наведеного аналізу можливо помітити, що найбільше переваг має бібліотека OpenCV. До того ж, дана бібліотека є доволі швидкодієюною в порівнянні з іншими програмними засобами. Також, OpenCV, містить натреновані класифікатори для детекції обличь, що дозволяє легко реалізувати алгоритми детекції обличь. Ще однією перевагою на користь OpenCV є підтримка широкого набору мов програмування, а

також наявність україномовної документації, що дещо спрощує розуміння бібліотеки та її основних компонентів.

Висновки до першого розділу

Окулографія (айтрекінг) – процес визначення точки, на яку спрямовується погляд чи рух ока відносно голови. Дана технологія широко використовується останні 10-15 років та застосовується майже у всіх сферах життєдіяльності.

Дана робота акцентує увагу використання окулографії у сфері навчання. Відстеження руху очей є ефективним інструментом для дослідження зорової уваги та соціальної взаємодії в різноманітних сферах навчання. Використовуючи цей метод, дослідники можуть розуміти, як різні аспекти можуть впливати на результат поглиблення інформації і як покращити процес навчання.

Аналізуючи результат дослідження ринку програмних та апаратних засобів окулографії можна зробити висновок, що існуючі аналоги мають значний недолік – вартість. Особливо це стосується сфери навчання. Не кожний навчальний заклад може дозволити собі придбати айтрекери для покращення процесу навчання. Особливо під час дистанційного навчання.

На сьогоднішній день, у зв'язку з поширенням епідемії коронавірусу Covid-19, закриваються навчальні заклади по всьому світу. В цей час надзвичайно зростає потреба у дистанційному навчанні, тому саме зараз процес покращення дистанційного навчання є найбільш актуальним. Студентам важко концентрувати увагу, коли вони засвоюють матеріал самостійно, без допомоги вчителя. Для покращення концентрації уваги необхідно дослідити поведінку студента під час навчання: на що студент звертає увагу, що залишається поза увагою, скільки часу він витрачає на засвоєння того чи іншого матеріалу.

Основним завданням даної роботи є дослідження розробки засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера з використанням апаратних засобів, які доступні кожному для покращення засвоєння інформації студентами під час дистанційного навчання.

Також необхідно дослідити відповідність отриманого результату відстеження погляду зі структурою інформації, яка відображена на моніторі екрану. Необхідно знайти область екрана, куди сфокусовано погляд респондента та час, який було витрачено на засвоєння інформації, яка спостерігається.

За результатами аналізу бібліотек комп'ютерного зору OpenCV, VXL та LTI, обрано бібліотеку OpenCV, оскільки вона є більш ефективною та продуктивною для реалізації засобів відстеження фокусу зору.

2 ОБҐРУНТУВАННЯ МЕТОДУ ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ

Технологія відстеження фокусу зору стала використовуватися у навчанні задля розуміння стратегії отримання нових знань у різноманітних умовах: від традиційного навчання до цифрового формату. Відстеження руху очей є ефективним інструментом для дослідження зорової уваги та соціальної взаємодії в різноманітних сферах навчання. Використовуючи цей метод, дослідники можуть розуміти, як різні аспекти можуть впливати на результат поглиблення інформації і як покращити процес навчання.

У ході розробки програмного засобу для дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК постає завдання вибору ефективного методу з використанням доступних апаратних засобів для досягнення поставленої мети.

2.1 Модель відстеження фокусу зору людини

Користувач ПК «user» спостерігає за інформацією на моніторі «screen», в той час відеокамера «web camera» записує пряму трансляцію зображення користувача. Основна ідея такої моделі – за допомогою програмного засобу, який встановлено на ПК маніпулювати поглядом очей для знаходження фокусу зору користувача.

На рис. 2.1 зображено модель відстеження фокусу зору людини під час роботи за комп'ютером з використанням однієї відеокамери, але можна використовувати і декілька.

Основні етапи процесу відстеження фокусу зору:

- **отримання відеокадрів:** система отримує відеокадри з веб-камери, яка записує пряму трансляцію зображення користувача;
- **обробка відеокадрів:** дозволяє отримати корисну інформацію з відеокадру, його необхідно обробити;
- **розпізнавання рис обличчя:** дозволяє мінімізувати область знаходження очей на кадрі, необхідно спочатку розпізнати обличчя;

- **розпізнавання очей:** дозволяє відстежувати розташування зіниці ока;
- **калібрування:** дозволяє досліджувати переміщення очей та отримати правильні і відтворювані експериментальні дані;
- **підрахунок координат фокусу зору:** дозволяє отримати координати фокусу зору на основі даних, отриманих під час виконання попередніх кроків.

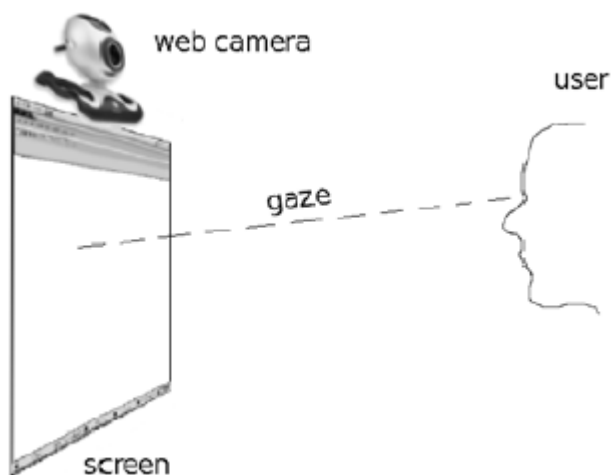


Рисунок 2.1 – Модель відстеження фокусу зору людини

2.2 Розпізнавання рис обличчя

Для розпізнавання рис обличчя було обрано активні моделі зовнішнього вигляду, так як вони призначені для точної локалізації антропометричних точок на зображенні обличчя та надаються бібліотекою комп'ютерного зору OpenCV. За допомогою активних моделей зовнішнього вигляду моделей можна легко локалізувати антропометричні точки області розміщення очей на обличчі.

Активні моделі зовнішнього вигляду – це статистичні моделі зображень, які шляхом різного роду деформацій можуть бути підігнані під реальне зображення. В активних моделях зовнішнього вигляду описуються два типи параметрів: параметри, пов'язані з формою (параметри форми), і параметри, пов'язані зі статистичною моделлю зображення або текстурою (параметри зовнішнього вигляду). Перед використанням модель повинна бути навчена на безлічі заздалегідь розмічених зображень. Кожна мітка

має свій номер і визначає характерну точку, яку повинна буде знаходити модель під час адаптації до нового зображення. Приклад подібної розмітки показаний на рис. 2.2.

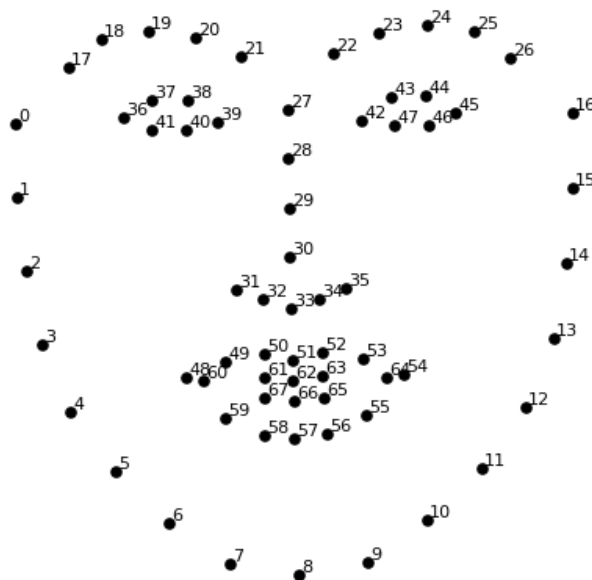


Рисунок 2.2 – Розмітка обличчя

У представленому прикладі на зображенні відзначені 68 міток, що утворюють форму моделі активного зовнішнього вигляду. Ця форма позначає зовнішній контур обличчя, контури рота, очей, носа, брів. Даний характер розмітки дозволяє в подальшому визначити різні параметри обличчя по зображенню, які можуть бути використані для подальшої обробки.

2.2.1 Процес розпізнавання рис обличчя

Для розпізнавання рис обличчя використовується бібліотека OpenCV.

Система отримує відеокادر з веб-камери та обробляє його за допомогою методом розпізнавання рисів обличчя. Обробка відеокадрів відбувається під час запису відеокамери прямої трансляції зображення користувача.

Під час ініціалізації методу розпізнавання рис обличчя, система використовує навчену модель зовнішнього вигляду, за допомогою якої можна спрогнозувати риси обличчя. Дана модель представляється у вигляді бази даних обличчя «shape_predictor_68_face_landmarks.dat». Існуюча база даних охоплює різні варіації

представлення обличчя: різні пози, освітлення, тощо [18]. Дана модель дає можливість розпізнати 68 міток, розмічених на обличчі (рис. 2.2). Приклад розпізнавання 68 міток на обличчі зображено на рис. 2.3.



Рисунок 2.3 – Приклади розпізнавання 68 міток на обличчі

Для розпізнавання обличчя на відеокадрі, його необхідно обробити, оскільки навчена модель зовнішнього вигляду працює лише з чорно-білими зображеннями.

Для збільшення продуктивності розпізнавання обличчя, необхідно підвищити інтенсивність зображення, задля цього використовується метод «Вирівнювання гістограм».

Результатом застосовування методу розпізнавання обличчя є масив координат міток, де кожна мітка має свій номер.

Отримавши масив координат міток активної моделі зовнішнього вигляду на відеокадрі, система знаходить антропометричні точки області розміщення очей на обличчі. Для лівого ока це мітки з номерами від 36 до 41. Для правого ока – від 42 до 47 (рис. 2.4).

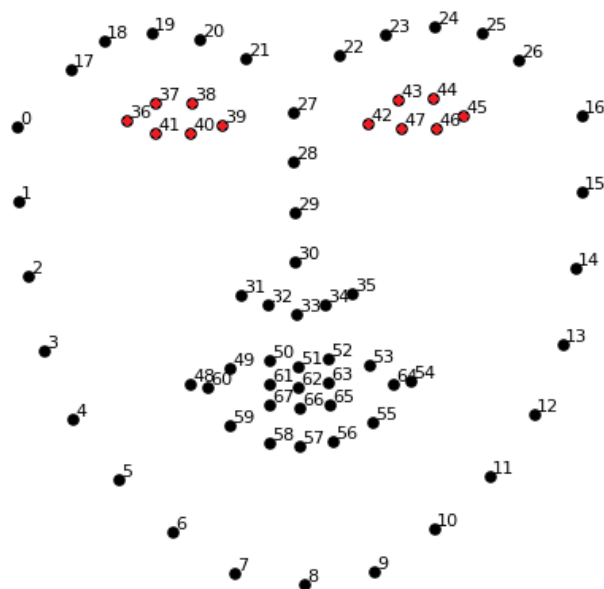


Рисунок 2.4 – Розмітка обличчя з виділеними антропометричними точками області розміщення очей

Приклади обробки відеокадрів з антропометричними точками області розміщення очей представлені на рис. 2.5.



Рисунок 2.5 – Обробка відеокадрів з антропометричними точками області розміщення очей

Не зважаючи на рух голови користувача, метод розпізнавання обличчя залишається придатним для вирішення поставленої задачі.

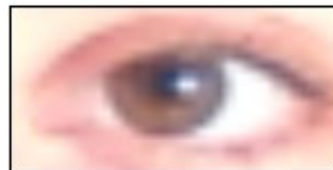
2.3 Розпізнавання очей

Наступним кроком системи відстеження фокусу зору є розпізнавання властивостей ока: координати центра ока та центра зіниці ока.

На даному етапі відстеження фокусу зору використовується лише частина відеокадру, на якому зображено лише область розміщення очей. Причому, кожне око обробляється окремо (рис. 2.6).



А



Б

Рисунок 2.6 – Приклад частин відеокадрів, які використовуються для розпізнавання очей: А – ліве око, Б – праве око

2.3.1 Знаходження координат центра ока

Нехай p_1, \dots, p_6 – антропометричні точки області розміщення очей. Кожна точка має координату абсцис (X) та ординат (Y). Координати центра ока (CoE) обчислюють за формулами 1.1 та 1.2:

$$CoE_x = \frac{p_{1x} + p_{4x}}{2}, \quad (2.1)$$

де p_{1x} та p_{4x} – координати абсцис.

$$CoE_y = \frac{p_{2y} + p_{3y} + p_{5y} + p_{6y}}{4}, \quad (2.2)$$

де p_{2y}, p_{3y}, p_{5y} та p_{6y} – координати ординат.

На рис. 2.7 схематично зображено метод знаходження координат центра ока.

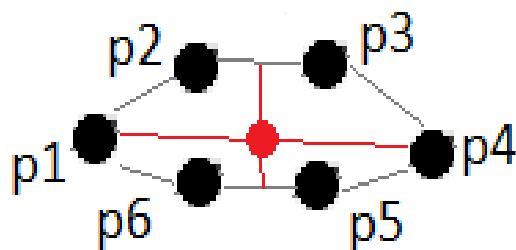


Рисунок 2.7 – Схема методу знаходження координат центра ока

2.3.2 Знаходження координат центру зіниці ока

Процес відстеження зіниці ока є одним з найскладніших етапів концепції відстеження фокусу зору. Саме від результату цього етапу залежить кінцевий результат, тому що аналізуючи переміщення зіниці ока можна відстежити координату фокусу зору.

Процес відстеження зіниці ока складається з декількох етапів.

На першому етапі проводиться обробка відеокадру. Зіниця – це найтемніша зона ока. Для того, щоб розпізнати зіницю, необхідно перевести кадр в градацію чорно-білих кольорів та підвищити інтенсивність кадру. На рис. 2.8 зображено кадр після обробки.



Рисунок 2.8 – Оброблений кадр правого ока

Для зменшення рівня шуму на кадрі використовується медіанна фільтрація. Коригуючи розмір апертури медіанного фільтра, можна отримати різний результат (рис. 2.9).

Далі використовується пороговий алгоритм для виділення найтемніших областей на зображенні. Кожен піксель на GrayScale зображенні має значення в діапазоні від 0 до 255, що означає його колір. Пороговий алгоритм дозволяє перетворити зображення в бінарний формат, тобто кожен піксель на зображенні може мати значення 0 – чорний або 255 – білий [19].

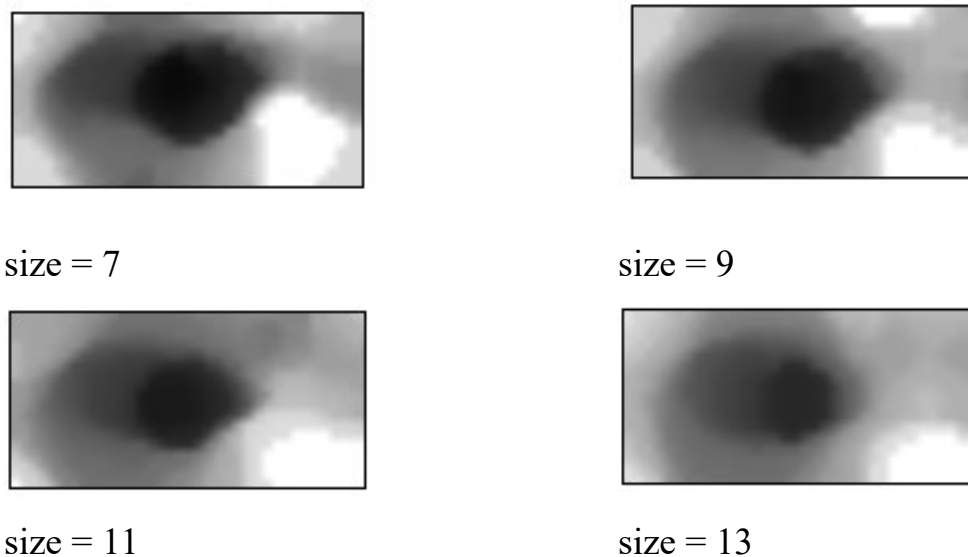


Рисунок 2.9 – Вплив розміру апертури на вихідне зображення

Пороговий алгоритм працює наступним чином:

$$Bin_{ij} = \begin{cases} 255, & I_{ij} \geq T \\ 0, & I_{ij} < T, \end{cases} \quad (2.3)$$

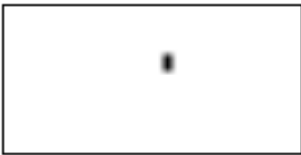



де Bin_{ij} – елемент бінарного зображення, I_{ij} – піксель на GrayScale зображення, T – порогове значення [20].

Результат алгоритму залежить від порогового значення (T) (рис. 2.10). В таблиці 2.1 представлено залежність результату від порогового значення.



Рисунок 2.10 – Фрагмент кадру правого ока до застосування порогового алгоритму

Таблиця 2.1 – Залежність результату алгоритму від обраного порогового значення (Т)

Порогове значення (Т)	Результат
T=10	
T=25	
T=40	
T=100	

В даному прикладі, успішним застосуванням порогового алгоритму вважається використання $T = 40$, оскільки на бінарному зображенні чітко відображається область зіниці ока.

Наступним етапом відстеження зіниці ока є знаходження контурів елементів на бінарному зображенні. Контур – крива, що з'єднує всі суцільні точки (вздовж межі), мають однаковий колір або інтенсивність. Для розпізнавання контурів на бінарному зображенні використовується алгоритм Сатоші Сузук [21], який надається бібліотекою комп'ютерного зору OpenCV. Приклади розпізнавання контурів елементів зображено в таблиці 2.2.


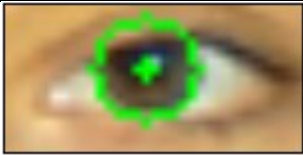
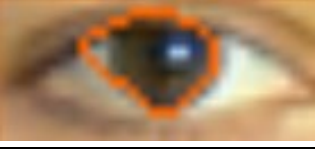

Отримавши масив контурів елементів, необхідно знайти найбільший контур, тому що він відповідає зіниці ока.

Таблиця 2.2 – Приклади розпізнавання контурів елементів

Зображення правого ока перед обробкою	Бінарне зображення ока	Зображення ока з відображення контурів елементів
		
		

Зіниця ока має форму кола. Для знаходження центра зіниці ока, необхідно підібрати таке коло, яке вписувалося би в область контуру елементу зіниці. Контур кола повинен наближуватися до контуру елементу зіниці. Для цього використовується алгоритм «Найменший огорожувальний диск» [22], який представлений бібліотекою OpenCV. Приклад знаходження радіуса кола зображено в таблиці 2.3

Таблиця 2.3 – Приклад знаходження радіуса кола

	Зображення ока з відображення контурів елементів	Зображення ока з відображення контура кола та центра зіниці ока
Праве око		
Ліве око		

2.4 Процес калібрування

Для підвищення точності розпізнавання фокусу зору необхідно ретельно дослідити рух очей користувача. Саме тому необхідно провести калібрування.

Калібрування – це процес, який полягає в послідовній демонстрації на екрані монітора точки з відомими координатами (X , Y) з синхронною реєстрацією спрямованого на неї погляду користувача з координатами (X , Y). На рис. 2.11 зображено приклад напрямку переміщення калібрувальної точки.

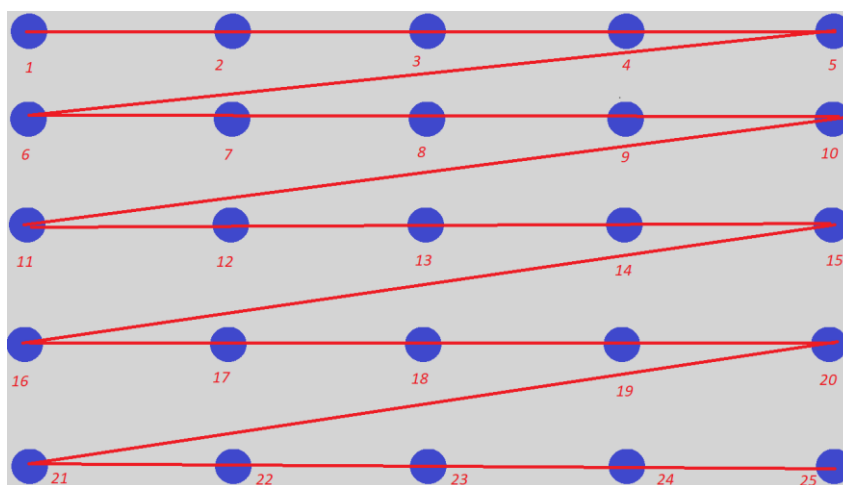


Рисунок 2.11 – Приклад напрямку переміщення калібрувальної точки

Під час калібрування використовується 25 позицій калібрувальної точки – 5 по горизонталі, та 5 по вертикалі, за якими спостерігає користувач. Точка змінює свою позицію через певний проміжок часу [23]. Паралельно з цим система записує координати зіниці очей, які відповідають кожній з позицій калібрувальної точки та зберігає дані у файлі csv формату.

Для уникнення дефектних даних, отриманих під час блимання очима, система відстежує процес блимання та не фіксує координати зіниці очей.

За активною моделлю зовнішнього вигляду, кожне око представлено шістьма антропометричними точками. Для лівого ока це мітки з номерами від 36 до 41. Для правого ока – від 42 до 47 (рис. 2.4). Приклад розміщення точок ока зображено на рис. 2.12.

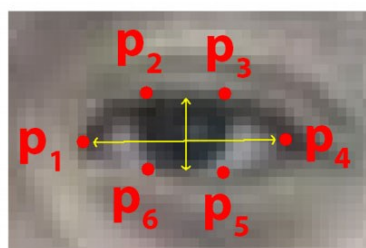


Рисунок 2.12 – Приклад розміщення точок ока

Існує залежність між шириною та висотою координат точок ока. Ґрунтуючись на дослідженнях Терези Сукупової та Яна Чеха «Виявлення моргання очей у режимі реального часу за допомогою орієнтирів на обличчі» [24], для знаходження співвідношення сторін ока використовується наступне рівняння:

$$EAR = \frac{|p_2 - p_6| + |p_3 - p_5|}{2|p_1 - p_4|}, \quad (2.4)$$

де EAR – співвідношення сторін ока, p_1, \dots, p_6 – антропометричні точки ока.

Співвідношення сторін ока є приблизно постійною величиною, поки око відкрите, але швидко падає до нуля, коли людина блимає.

При відкритому оці, величина EAR є великою і відносно постійною з часом. Однак, як тільки людина блимає, EAR ока різко зменшується, наближаючись до нуля.

На рис. 2.13 зображено графік співвідношення сторін ока у часі для відеокліпу. Протягом певного часу співвідношення сторін ока постійне, потім швидко падає майже до нуля, а потім знову збільшується, що вказує на те, що відбулося одне моргання.

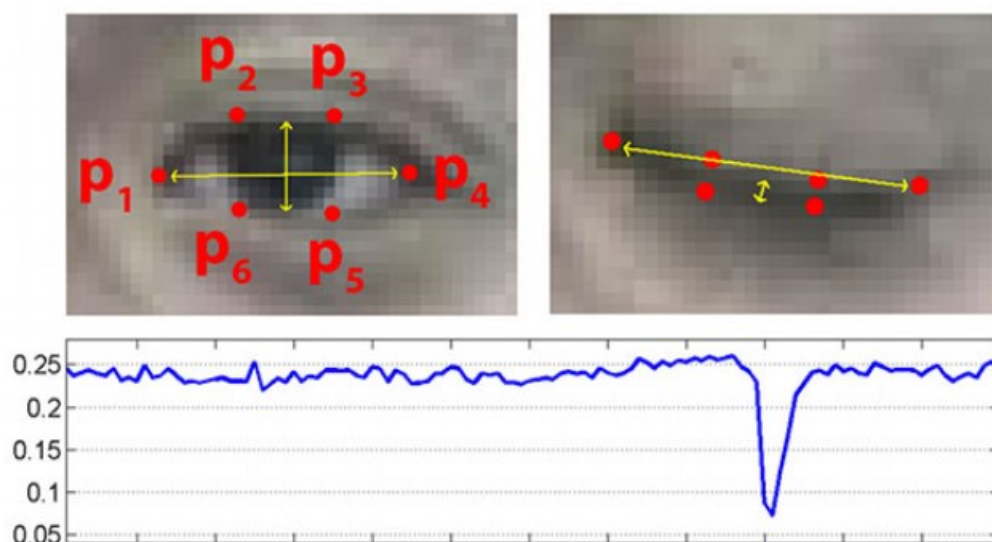


Рисунок 2.13 – Залежність співвідношення сторін очей від стану ока

Система відстеження фокусу зору використовує рівняння (2.4) задля фіксації блимання очей. У системі задано порогове значення $EAR = 0.15$. Якщо отриманий результат співвідношення сторін очей під час обробки відеокадру є нижчим або дорівнює установленому пороговому значенню, то система фіксує блимання очей та зупиняє обробку кадру.

Результатом калібровки є співвідношення координат зіниці ока та позицій калібрувальної точки. Наступним кроком є обробка отриманих даних – знаходження лінійної регресійної залежності координати зіниці ока від позиції калібрувальної точки на екрані:

$$y = x\beta_0 + \beta_1, \quad (2.5)$$

де y – координата зіниці ока, x – координата позиції калібрувальної точки, β_0 та β_1 – невідомі параметри.

Для кожної осі лінійна залежність підраховується окремо.

Для розрахунку параметрів моделі лінійної регресії застосовується метод найменших квадратів [25]:

$$\beta_0 = \frac{1}{M} \sum y_i - \frac{\beta_1}{M} \sum x_i, \quad \beta_1 = \frac{M \sum x_i y_i - \sum x_i \sum y_i}{M \sum x_i^2 - (\sum x_i)^2}, \quad (2.6)$$

де M – розмір вибірки даних (y_i, x_i) .

Використовуючи знайдені параметри лінійної регресії, система відстеження фокусу зору підраховує координати фокусу зору.

2.5 Підрахунок координат фокусу зору

Пошук координат фокусу зору є останнім етапом процесу відстеження фокусу зору. Щоб виконати підрахунки обчислення координат фокусу зору, перед цим необхідно визначити місце розташування очей на кадрі, їх властивості: координати центра ока (CoE) та центр зіниці ока (CoI), а також провести калібрування, задля отримання правильних і відтворюваних експериментальних даних.

Наступним кроком є визначення відстані зміщення центра зіниці відповідно центру ока.

Для лівого ока відстань зміщення підраховується наступним чином:

$$L_{LX} = CoE_{LX} - CoI_{LX}, \quad (2.7)$$

де L_{LX} – відстань зміщення для лівого ока по осі абсцис,

CoE_{LX} – координата центра лівого ока по осі абсцис,

та CoI_{LX} – координата центра зіниці лівого ока по осі абсцис.

$$L_{Ly} = CoE_{Ly} - CoI_{Ly}, \quad (2.8)$$

де L_{Ly} – відстань зміщення для лівого ока по осі ординат,

CoE_{Ly} – координата центра лівого ока по осі ординат,

та CoI_{Ly} – координата центра зіниці лівого ока по осі ординат.

Для правого ока відстань зміщення L_R по осі абсцис і ординат підраховується аналогічним чином за рівняннями (2.7) та (2.8) відповідно.

Отримавши відстань зміщення для кожного ока, підраховується середній показник зміщення:

$$L_{AverX} = (L_{LX} + L_{RX})/2, \quad (2.9)$$

де L_{AverX} – середній показник зміщення по осі абсцис.

$$L_{AverY} = (L_{LY} + L_{RY})/2, \quad (2.10)$$

де L_{AverY} – середній показник зміщення по осі ординат.

Далі безпосередньо знаходиться точка фокусу зору:

$$PoG_X = (L_{AverX} - \beta_{0X})/\beta_{1X}, \quad (2.11)$$

де PoG_X – координата фокусу зору по осі абсцис,

β_{0X}, β_{1X} – коефіцієнти лінійної регресії, отримані під час обробки калібрувальних даних по осі абсцис.

$$PoG_Y = (L_{AverY} - \beta_{0Y}) / \beta_{1Y}, \quad (2.12)$$

де PoG_Y – координата фокусу зору по осі ординат,

β_{0Y}, β_{1Y} – коефіцієнти лінійної регресії, отримані під час обробки калібрувальних даних по осі ординат.

Якщо отримані координати точки фокусу зору виходять за межі координат монітору ($0 > PoG_X > W$ та $0 > PoG_Y > H$, де W та H – ширина та висота монітору у пікселях відповідно), тоді їм надається номінальне значення меж координат монітору.

2.6 Оцінка похибки знаходження координат фокусу зору

Для оцінки похибки знаходження координат фокусу зору проводиться ще один процес калібрування, під час якого система записує безпосередньо розраховані координати фокусу зору та відповідні координати калібрувальної точки та зберігає дані у файлі csv формату. По завершенню роботи системи відбувається аналіз зібраних даних та підраховується похибка.

Похибка відображає відхил розрахованих координат фокусу зору (PoG) від координат калібрувальної точки (PoC) відповідно до розмірів екрану монітора. Похибку, виражену у відсотках, обчислюють за наступними формулами:

$$Error_x = \frac{(|PoG_X - PoC_X|)}{W} \times 100\%, \quad (2.13)$$

де $Error_x$ – похибка по осі абсцис,

PoG_X – розрахована координата фокусу зору по осі абсцис,

PoC_X – координата калібрувальної точки по осі абсцис,

W – ширина монітора.

$$Error_Y = \frac{(|PoG_Y - PoC_Y|)}{H} \times 100\%, \quad (2.14)$$

де $Error_Y$ – похибка по осі ординат,

PoG_Y – розрахована координата фокусу зору по осі ординат,

PoC_Y – координата калібрувальної точки по осі ординат,

H – висота монітора.

2.7 Пошук відповідності фокусу зору зі структурою інформації на моніторі

Для дослідження відповідності фокусу зору зі структурою інформації на моніторі, необхідно знати місце розташування конкретного інформаційного контенту на екрані. Для вирішення цього питання створюється спеціальний файл, в який записуються дані про розміщення елементів інформаційного контенту: зображення, формули, абзаци, заголовки, тощо. Дані в файлі представляються у вигляді json формату. Пошук даних відбувається вручну.

Json формат інформаційного контенту містить два об'єкти – пар назва/значення:

- «header» – заголовок, який містить загальні дані про інформаційний контент:
 - «title» – назва інформаційного контенту;
 - «type» – вид інформаційного контенту (наприклад, стаття, журнал, книга, тощо);
 - «link» – джерело, звідки був взятий інформаційний контент;
- «content» – масив елементів інформаційного контенту. Кожний елемент містить наступні об'єкти:
 - «item» – вид елементу інформаційного контенту (наприклад, зображення, абзац, формула, тощо);
 - «pos» – область розміщення елементу інформаційного контенту;
 - «info» – додаткові дані про елемент інформаційного контенту.

Приклад даних інформаційного контенту в json форматі:

```
{
  "header":
  {
    "title": "Принцип построения алгоритмов быстрого преобразования Фурье",
    "type": "article",
    "link": "http://ru.dsplib.org/content/fft_introduction/fft_introduction.html"
  },
  "content":
  [
    {
      "item": "head",
      "pos": [ 95, 7, 926, 53 ],
      "info": "Наименование статьи"
    },
    {
      "item": "text",
      "pos": [ 0, 1055, 1024, 131 ]
    },
    {
      "item": "list",
      "pos": [ 0, 3996, 1104, 462 ],
      "info": "Список литературы"
    },
    {
      "item": "image",
      "pos": [ 863, 128, 238, 328 ],
      "info": "Портрет Д.Тьюки"
    },
    {
      "item": "formula",
      "pos": [ 145, 929, 563, 106 ]
    }
  ]
}
```

Створивши файл з даними про розміщення елементів інформаційного контенту, система зчитує його та шукає відповідності фокусу зору користувача зі структурою інформації на моніторі. Система відстежує, на який елемент інформаційного контенту

сфокусовано погляд користувача у певний період часу та записує отримані дані в csv файл.

По завершенню роботи системи, дослідник може зчитати зібрані дані системою та провести дослідження поведінки користувача під час роботи з інформаційного контентом на моніторі. Дослідження можуть вказати на що користувач звертав більше уваги, що залишилося поза увагою, скільки часу він витратив на розуміння того чи іншого матеріалу.

2.8 Системні обмеження для відстеження фокусу зору

Для досягнення кращої продуктивності відстеження фокусу зору, необхідно дотримуватися деяких системних обмежень:

- відеокамера повинна розташовуватися навпроти користувача, на відстані 50 см, щоб забезпечити найкращий огляд обличчя людини;
- необхідно забезпечити рівномірно освітлене приміщення, адже зміна освітлення може знизити вірогідність розпізнавання;
- користувач повинен знаходитися навпроти центру екрана, щоб відеокамера фіксувала повністю все обличчя;
- користувач не повинен використовувати окуляри, адже це може погіршити процес розпізнавання;
- користувач не повинен вертати головою під час роботи за комп'ютером, бо це може погіршити розпізнавання зіниці.

2.9 Опис апаратних та програмних засобів

Для функціонування системи відстеження відповідності фокусу зору зі структурою інформації на моніторі, необхідно використовувати наступні засоби:

- апаратні засоби:
 - ноутбук з процесором Intel Core i5-3230M, RAM – 4Gb, відеокартою – NVIDIA GeForce GT 740M та монітором з роздільною здатністю 1366x768;

- відеокамера з роздільною здатністю 1280x720 (в даному випадку використовувалася відеокамера, вбудована в ноутбук);
- додаткова зовнішня відеокамера з роздільною здатністю 1280x1024;
- програмні засоби:
 - операційна система Windows 10.

Висновки до другого розділу

Таким чином, метод відстеження відповідності фокусу зору зі структурою інформації на моніторі відбувається в декілька етапів: отримання та обробка відеокадрів, розпізнавання рис обличчя та очей на відеокадрі, калібрування даних, підрахунок координат фокусу зору.

Було обрано метод «Активні моделі зовнішнього вигляду» для розпізнавання рис обличчя, оскільки він забезпечує локалізацію антропометричних точок на зображенні обличчя, в тому числі точки області розміщення очей.

Використовуючи процес калібрування, система відстеження фокусу зору підвищує точність розпізнавання. Система записує координати зіниці очей, відповідно відображеній позиції калібрувальної точки, а далі шукає залежність між цими даними. Для цього використовується модель лінійної регресії з застосуванням методу найменших квадратів. Отримавши залежність, система використовує ці дані для підрахунку координат фокусу зору.

Система відстеження фокусу зору має деякі обмежень, яких необхідно дотримуватися, задля досягнення кращої продуктивності відстеження фокусу зору.

Для дослідження відповідності фокусу зору зі структурою інформації на моніторі, система використовує спеціальний файл, який містить дані про розміщення елементів інформаційного контенту та відстежує, на який елемент інформаційного контенту сфокусовано погляд користувача у певний період часу. Отримані дані записуються у csv файл, які потім можуть бути використані при дослідженні поведінки користувача під час роботи з інформаційним контентом на моніторі.

3 РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ

3.1 Зовнішнє проектування

3.1.1 Формалізація задачі

Формалізація задачі на рівні зовнішнього проектування наведена у вигляді діаграми варіантів використання (рис. 3.1). Діаграма варіантів використання – це граф спеціального вигляду, який є графічною нотацією для представлення конкретних варіантів використання, акторів, можливо деяких інтерфейсів, і відносин між цими елементами [26].

В якості акторів системи відстеження відповідності фокусу зору зі структурою інформації на моніторі виступають декілька суб'єктів: користувач, дослідник, програмне забезпечення для відстеження відповідності фокусу зору зі структурою інформації на моніторі та відеокамера. Кожен з цих акторів має свою роль у системі.

Адміністратор займається наступними задачами:

- налаштування системи для покращення розпізнавання зіниці ока: вибір значення апертури для медіанного фільтру та порогового значення для бінарізації зображення;
- налаштування процесу калібрування: вибір форми, кольору та розміру калібрувальної точки, визначення кількості позицій калібрувальної точки по горизонталі та вертикалі та встановлення часового періоду зміни позиції калібрувальної точки;
- налаштування інформаційного контенту за яким спостерігатиме користувач: вибір файлів з інформаційним контентом та даними про розміщення елементів інформаційного контенту;

- аналіз результуючих даних системи відстеження відповідності фокусу зору зі структурою інформації на моніторі, задля дослідження поведінки користувача під час роботи з інформаційним контентом на моніторі.

Основним завданням користувача є спостереження за позицією калібрувальної точки під час процесу калібрування та під час розрахунку похибки відстеження координат фокусу зору, а також робота з інформаційним контентом, зображеним на моніторі.

Відеокамера займається безпосередньо записом зображення користувача в прямій трансляції.

Основним та найважливішим актором системи є програмне забезпечення для відстеження відповідності фокусу зору зі структурою інформації на моніторі. Програмне забезпечення займається наступними задачами:

- отримання кадрів з відеокамери під час запису користувача при роботі з ПК;
- обробка отриманих відеокadrів;
- розпізнавання рис обличчя на кадрі;
- знаходження антропометричних точок розташування очей на обличчі;
- знаходження координат центру ока;
- знаходження координат центру зіниці ока;
- проведення процесу калібрування;
- відстеження процесу блимання очей;
- пошук залежності координати зіниці ока від позиції калібрувальної точки;
- підрахунок координати фокусу зору;
- оцінка похибки знаходження координати фокусу зору;
- пошук відповідності фокусу зору зі структурою інформації на моніторі.

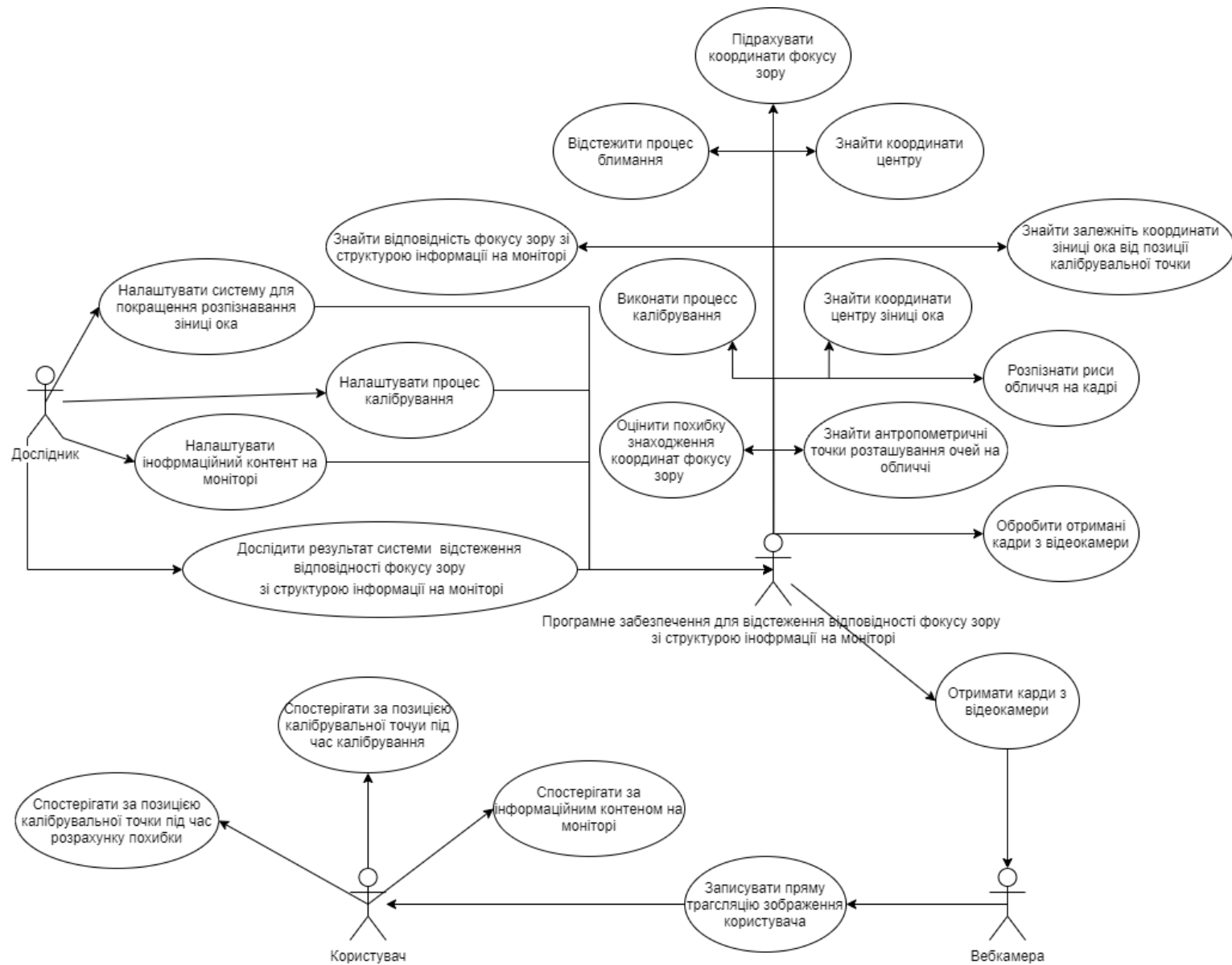


Рисунок 3.1 – Діаграма варіантів використання

3.2 Базова архітектура системи

У процесі проектування було виконано декомпозицію модулів системи відстеження відповідності фокусу зору зі структурою інформації на моніторі на три рівні згідно з моделлю «Модель-вигляд-контролер» (MVC) [29], результати якої наведені на рис. 3.2.

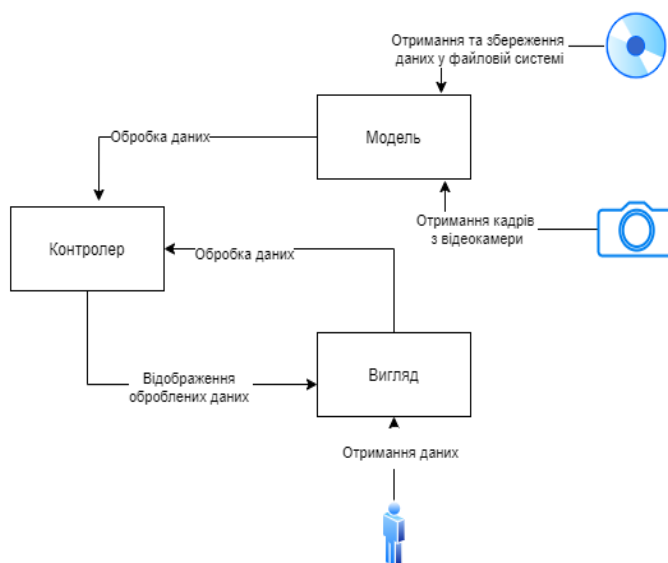


Рисунок 3.2 – Модель «Model-View-Controller» системи відстеження відповідності фокусу зору зі структурою інформації на моніторі

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (вигляду). За рахунок такого поділу підвищується можливість повторного використання коду.

У рамках архітектурного шаблону MVC програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами:

- модель (Model) – відповідає за збереження та отримання даних з файлової системи та відеокамери. За допомогою цього компоненту система отримує кадри з відеокамери, зберігає результат калібрування у файл, зчитує дані за файлу для відображення інформаційного контенту, тощо;

- вигляд (View) – відповідає за представлення даних користувачеві (наприклад, відображення процесу калібрування, інформаційного контенту, тощо). Компонент представлений у вигляді графічного інтерфейсу;
- контролер (Controller) – відповідає за бізнес-логіку. Контролер виконує основну частину роботи системи відстеження відповідності фокусу зору зі структурою інформації на моніторі. Він отримує дані від компоненту Model, обробляє їх та відправляє у компонент View для відображення користувачеві. Також контролер отримує сигнали у вигляді реакції на дії користувача (натискання кнопки, введення даних у текстове поле) та обробляє їх.

Для реалізації системи відстеження відповідності фокусу зору зі структурою інформації на моніторі обрана об'єктно-орієнтована та функціональна парадигми програмування [30].

3.3 Внутрішнє проектування

3.3.1 Вибір інструментальних засобів розробки

Для розробки системи відстеження відповідності фокусу зору зі структурою інформації на моніторі обрано мову програмування Python, так як він є зручним для розуміння, широко використовується для математичних обчислень, для роботи з великими даними та для машинного навчання. Середовище розробки обрано PyCharm.

Для відстеження фокусу зору обрана бібліотека комп'ютерного зору OpenCV. Бібліотека OpenCV є найпопулярнішою і розвиненою в області обробки зображень і комп'ютерного зору, має базові методи обробки зображення, методи геометричного перетворення, перетворення кольірних просторів, машинного навчання, виявлення об'єктів на зображенні. Вихідний код бібліотеки написаний на Python.

Для реалізації графічного інтерфейсу обрана бібліотека PyQt5. Для розробки дизайну графічного інтерфейсу обрано Qt Designer середовище розробки.

3.3.2 Ієрархія та взаємодія модулів системи

На рис. 3.3 представлено схему зображення взаємодії модулів системи відстеження відповідності фокусу зору зі структурою інформації на моніторі

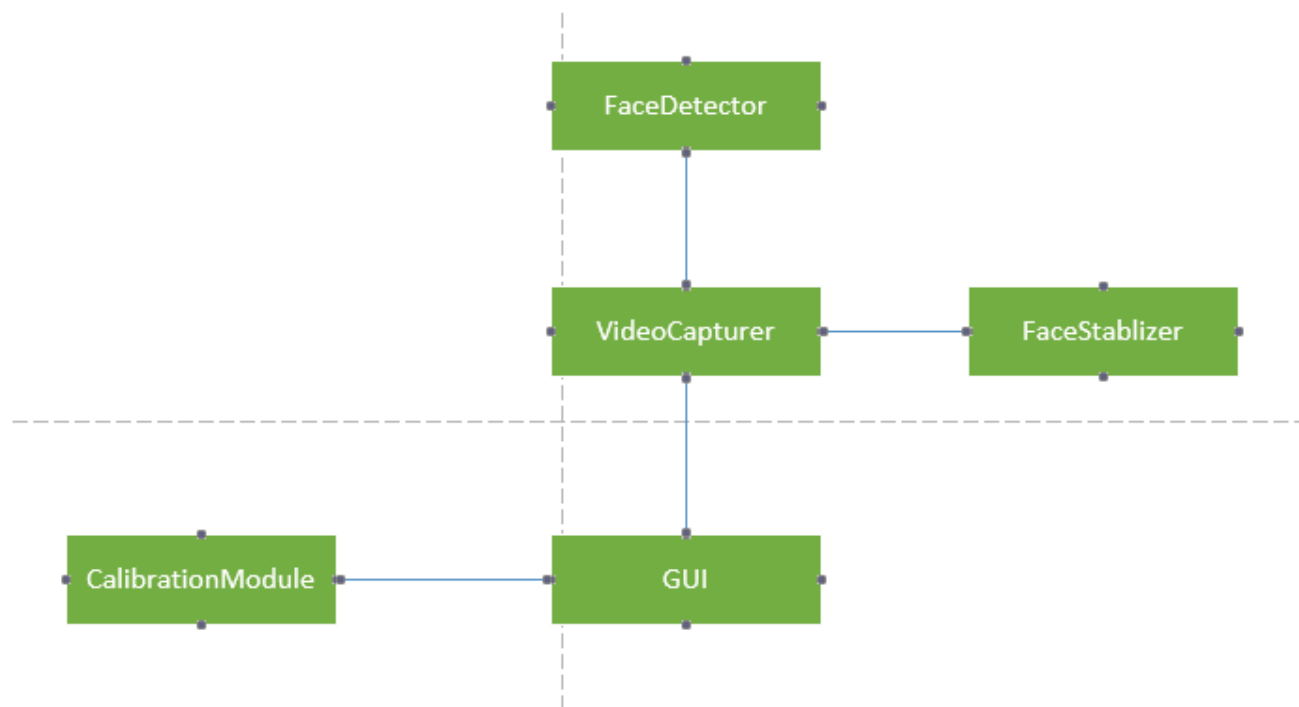


Рисунок 3.3 – Схема взаємодії модулів системи

Пакет «FaceDetector» (рис. 3.4) дозволяє розпізнати образ обличчя на відеокадрі та містить наступні класи:

- FaceDetector – відстеження риси обличчя;
- Face – структура для представлення даних про риси обличчя;
- Eye – структура для представлення даних про око;
- Pupil – структура для представлення даних про зіницю ока;
- EyeDetector – відстеження даних про око на відеофрагменті;
- PupilDetector – відстеження даних про зіницю на відеофрагменті.

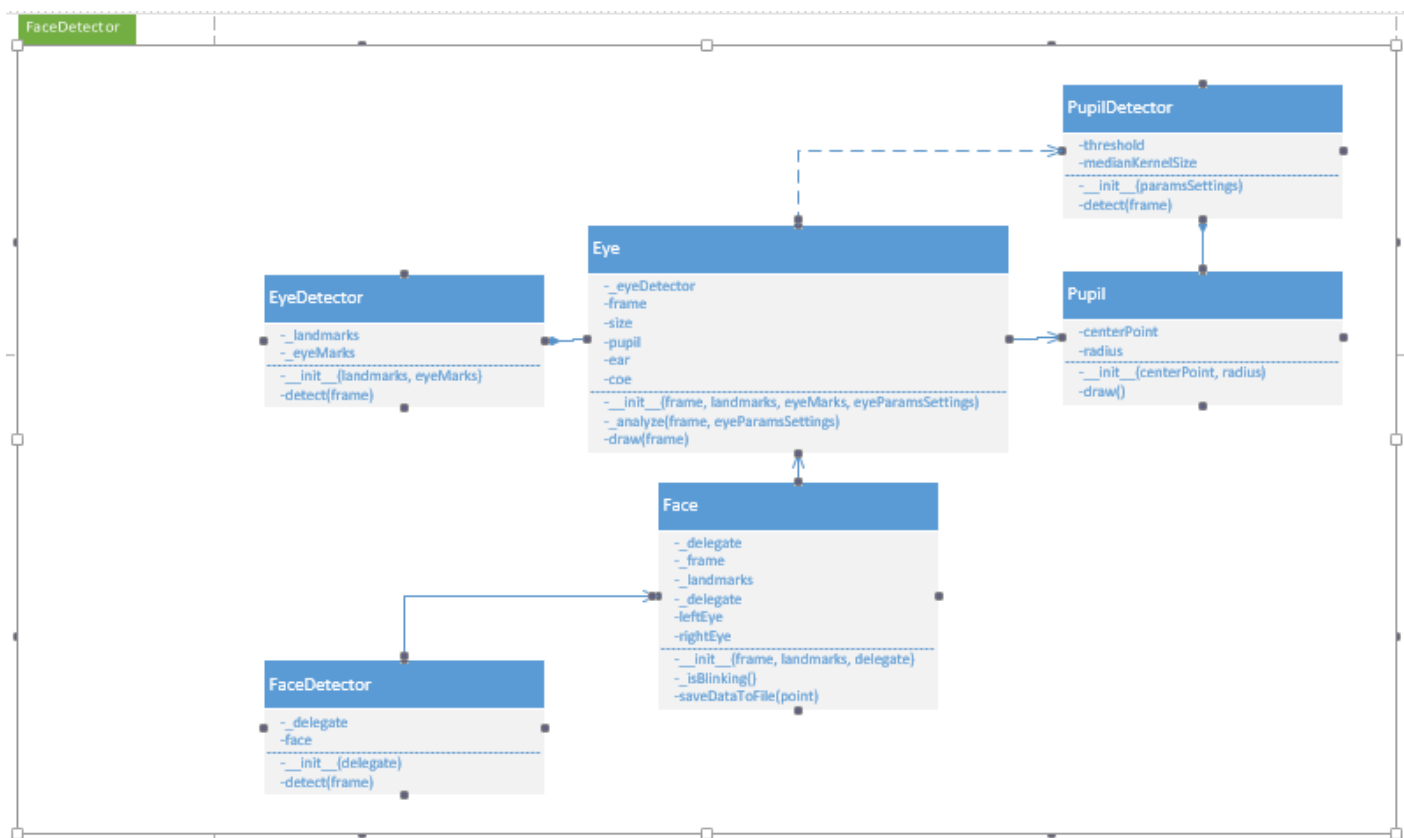


Рисунок 3.4 – Діаграма класів пакета «FaceDetector»

Пакет «FaceStabilizer» (рис. 3.5) дозволяє стабілізувати отримані дані про розпізнані образи обличчя для кількох кадрів (кількість кадрів дорівнює 4), щоб позбутися шумів під час підрахунку координат фокусу зору.

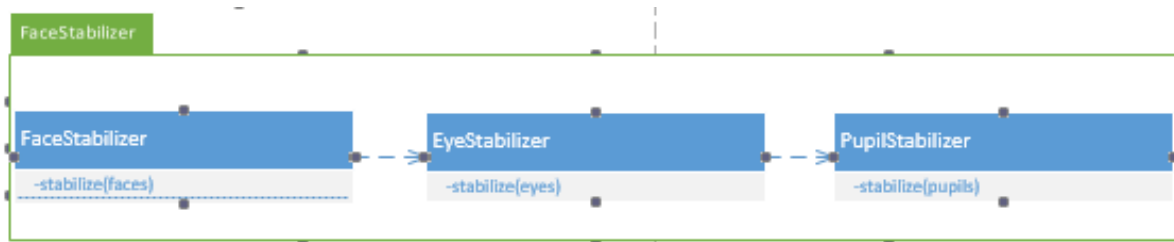


Рисунок 3.5 – Діаграма класів пакета «FaceStabilizer»

Пакет «PointGazeDetector» (рис. 3.6) дозволяє відстежити фокус зору під час роботи з інформаційним контентом.

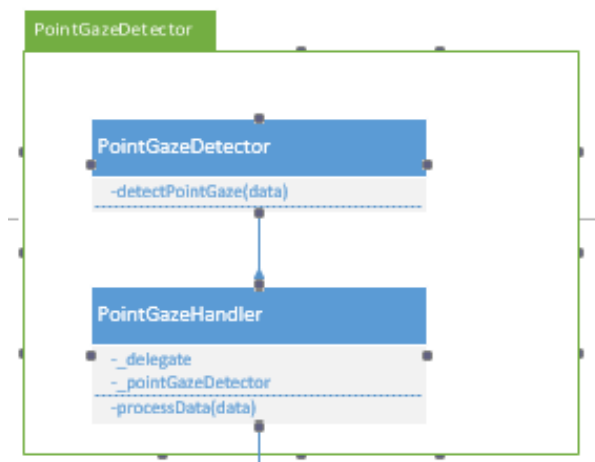


Рисунок 3.6 – Діаграма класів пакета «PointGazeDetector»

Пакет «VideoCapturer» (рис. 3.7) дозволяє отримувати відеокадри з відеокамер.

Клас FrameCapturer дозволяє захвачувати кадри з декількох відеокамер одночасно. Клас VideoCapturer дозволяє захвачувати кадри з певної відеокамери. Кожний отриманий кадр піддається обробці за допомогою FrameAnalyzer.

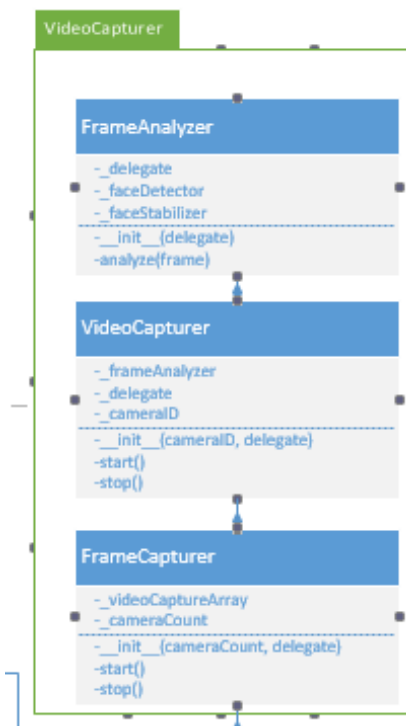


Рисунок 3.7 – Діаграма класів пакета «VideoCapturer»

Пакет «CalibrationModule» (рис. 3.8) відповідає за процес калібрування.

Базовим класом розробки функціоналу процесу калібрування є CalibrationModule. Клас CalibrationSettings представляє собою опції налаштування процесу калібрування. Опції налаштування зчитуються з файлу. Клас CalibrationWidget представляє собою віджет, який з'являється під час виконання процесу калібрування та відображає позиції калібрувальної точки. Після завершення калібрування, отримані дані обробляються за допомогою класу DataAnalyzer.

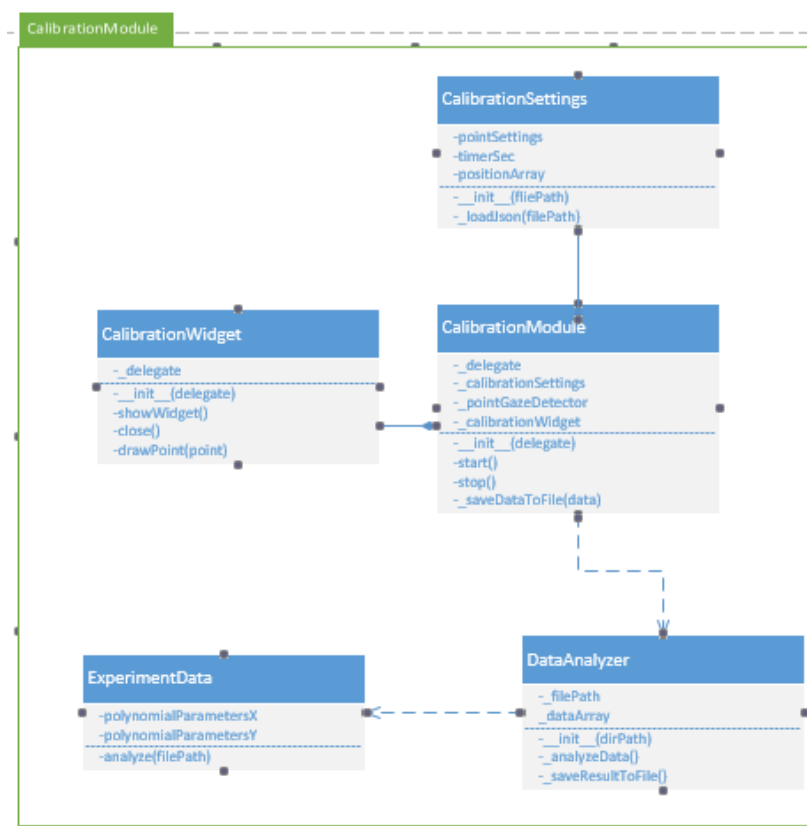


Рисунок 3.8 – Діаграма класів пакета «CalibrationModule»

Пакет «GUI» (рис. 3.9) відповідає за відображення графічного інтерфейсу та обробку сигналів на дії користувача.

Базовим класом ієрархії є MainWindow – клас, який відповідає за обробку сигналів графічного інтерфейсу та є спадкоємцем класу QMainWindow. MainWindow є початковою точкою роботи системи відстеження відповідності фокусу зору зі структурою інформації на моніторі. Для полегшення обробки сигналів графічного інтерфейсу вікон графічного інтерфейсу, було розроблено наступні класи:

LeftEyeSettings, RightEyeSettings, CalibrationTab та InformationContentTab. Кожен клас містить атрибут delegate – посилання на клас MainWindow.

Класи LeftEyeSettings та RightEyeSettings відповідають за обробку сигналів налаштування відстеження лівої та правої зіниці ока. Клас EyeSettings є базовим класом для LeftEyeSettings та RightEyeSettings.

Клас CalibrationTab відповідає за обробку сигналів налаштування процесу калібрування. Клас InformationContentTab відповідає за обробку сигналів налаштування процесу відстеження відповідності фокусу зору зі структурою інформації. За відображення інформаційного конфетну на моніторі відповідає InformationContextWidget.

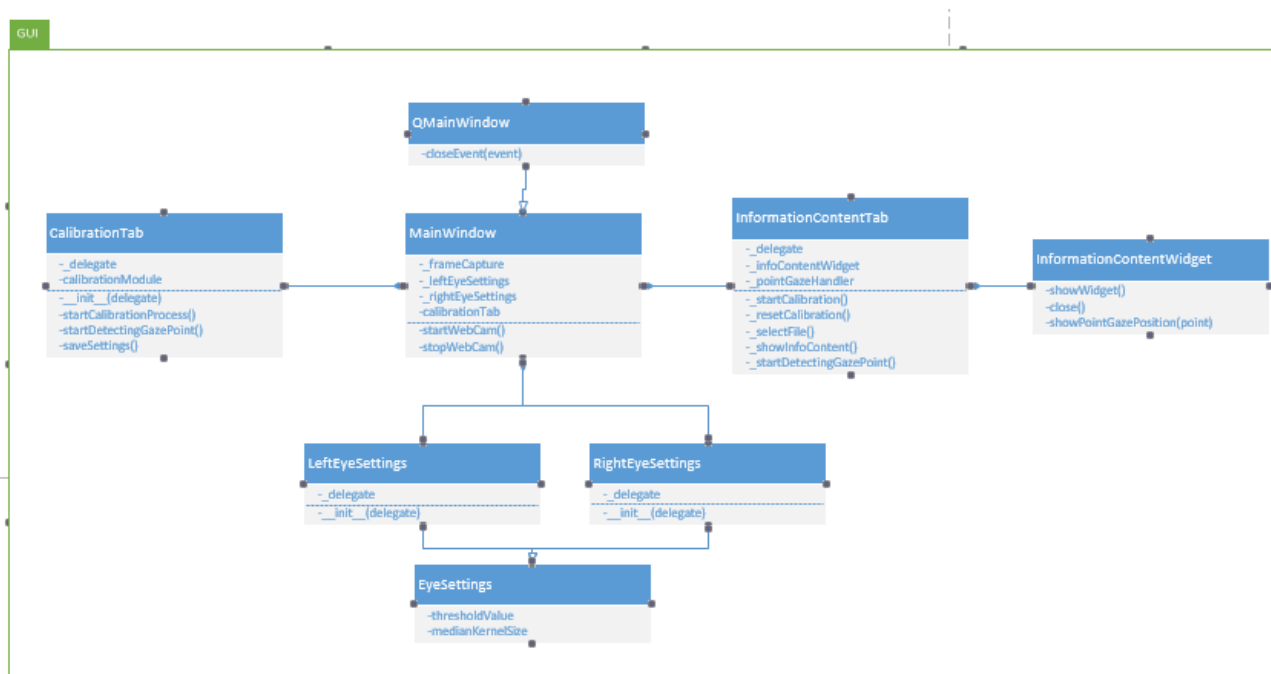


Рисунок 3.9 – Діаграма класів пакета «GUI»

3.3.3 Використані принципи проектування

Для проектування внутрішньої структури системи відстеження відповідності фокусу зору зі структурою інформації на моніторі використовувалися такі принципи об'єктно-орієнтованого проектування:

- принцип інкапсуляції: прикладом є атрибути класів, ім'я яких починається з префіксу «_» (рис. 3.4-3.9);
- принцип успадкування: прикладом є класи LeftEyeSettings та RightEyeSettings, які унаслідуються від EyeSettings (рис. 3.9);
- принцип поліморфізм: прикладом є перевантаження деяких системних методів QMainWindow класом MainWindow для впровадження необхідної логіки (рис. 3.9);
- принцип єдиного обов'язку: прикладом є представлені класи на діаграмі класів, де кожен клас системи має лише один обов'язок (рис. 3.4-3.9).

3.4 Розробка інтерфейсу користувача

Програмне забезпечення відстеження відповідності фокусу зору зі структурою інформації на моніторі має назву **Eye Tracking**. Інтерфейс користувача програмного забезпечення було поділено на декілька вікон графічного інтерфейсу, задля покращення розуміння його використання.

«Головне вікно» – вікно, яке з'являється після запуску програми (рис. 3.10, 3.11). Воно дозволяє обрати кількість відеокамер – одну або дві, які будуть використовуватися для запису користувача та розпочати перехват відеокадрів з камер, натиснувши на кнопку «Start Video». Для візуального відображення контурів очей та зіниць очей на відеокадрах, існують спеціальні опції «Show eye contours», «Show eye pupils». Для зупинки перехвату відеокадрів з камер використовується кнопка «Stop Video».

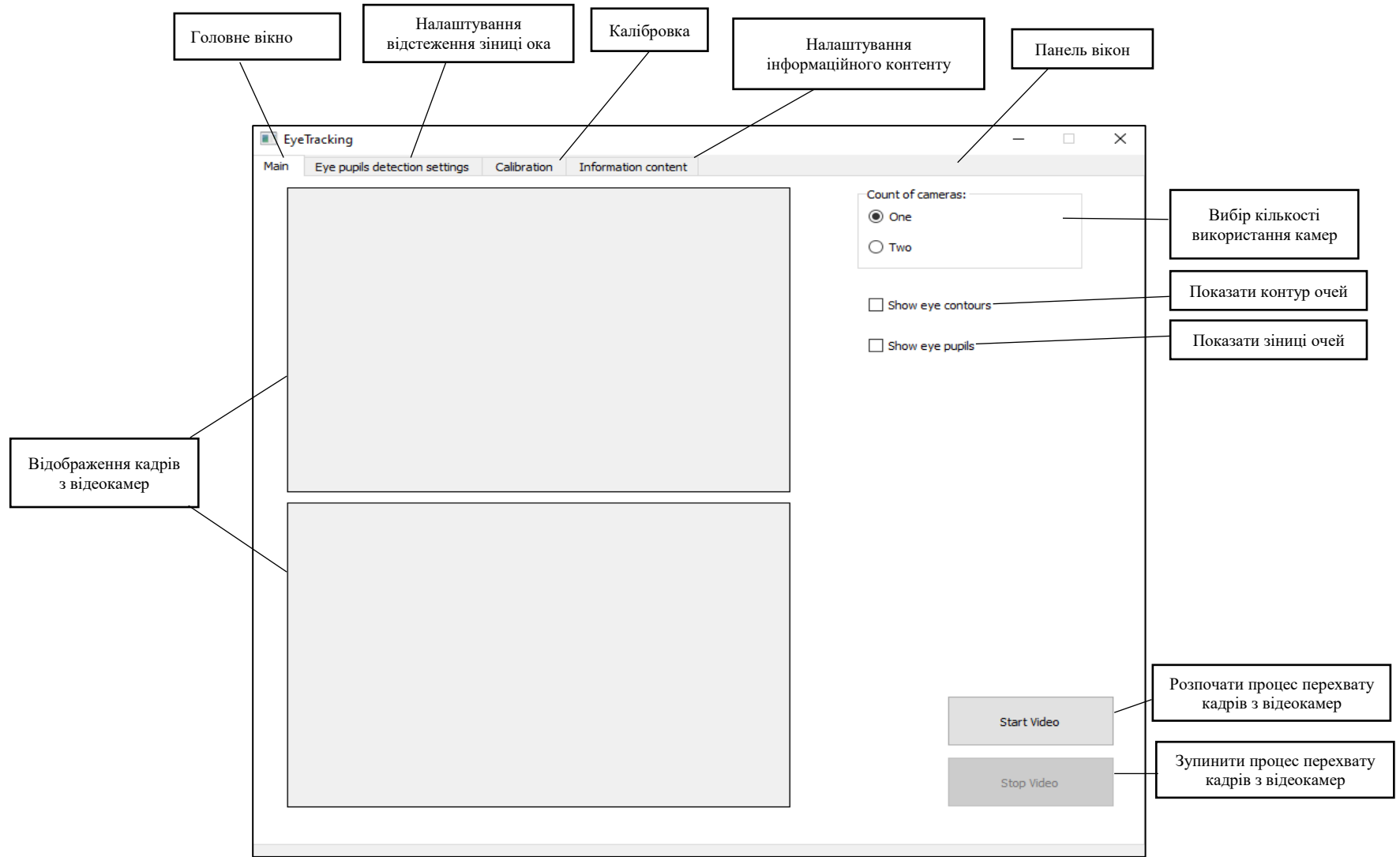


Рисунок 3.10 – Опис елементів головного вікна програми

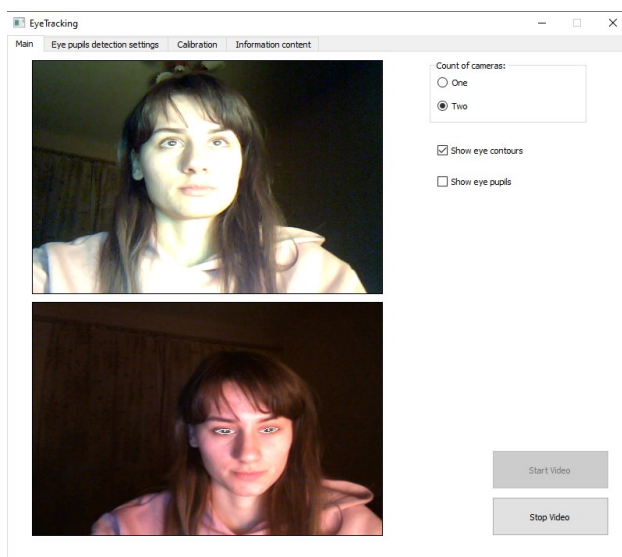


Рисунок 3.11 – Вигляд головного вікна під час обробки відеокадрів з двох камер

Вікно «Налаштування відстеження зіниць очей» (рис. 3.12, 3.13) містить два підвікна, які дозволяють налаштувати процес відстеження зіниці ока для лівого та правого ока. У кожному вікні надається можливість встановити розмір апертури медіанного фільтру та порогове значення порогового фільтру та дослідити поетапний процес відстеження зіниці ока.

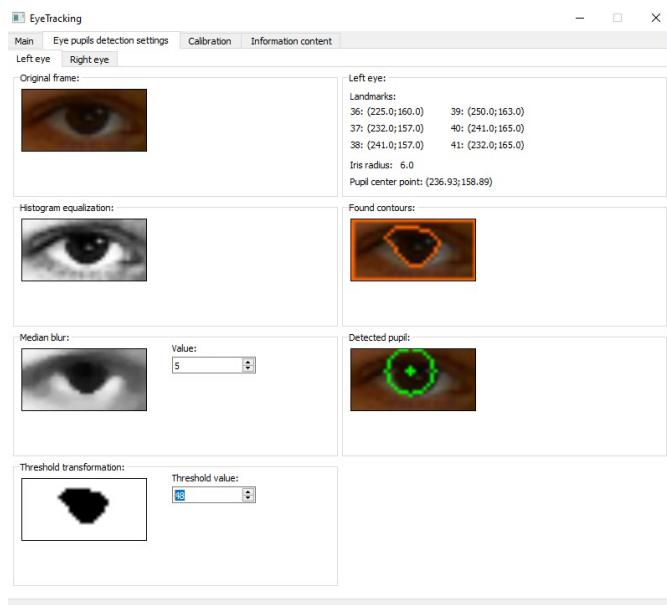


Рисунок 3.12 – Вікно «Налаштування відстеження зіниць очей» під час налаштування процесу відстеження зіниці для лівого ока

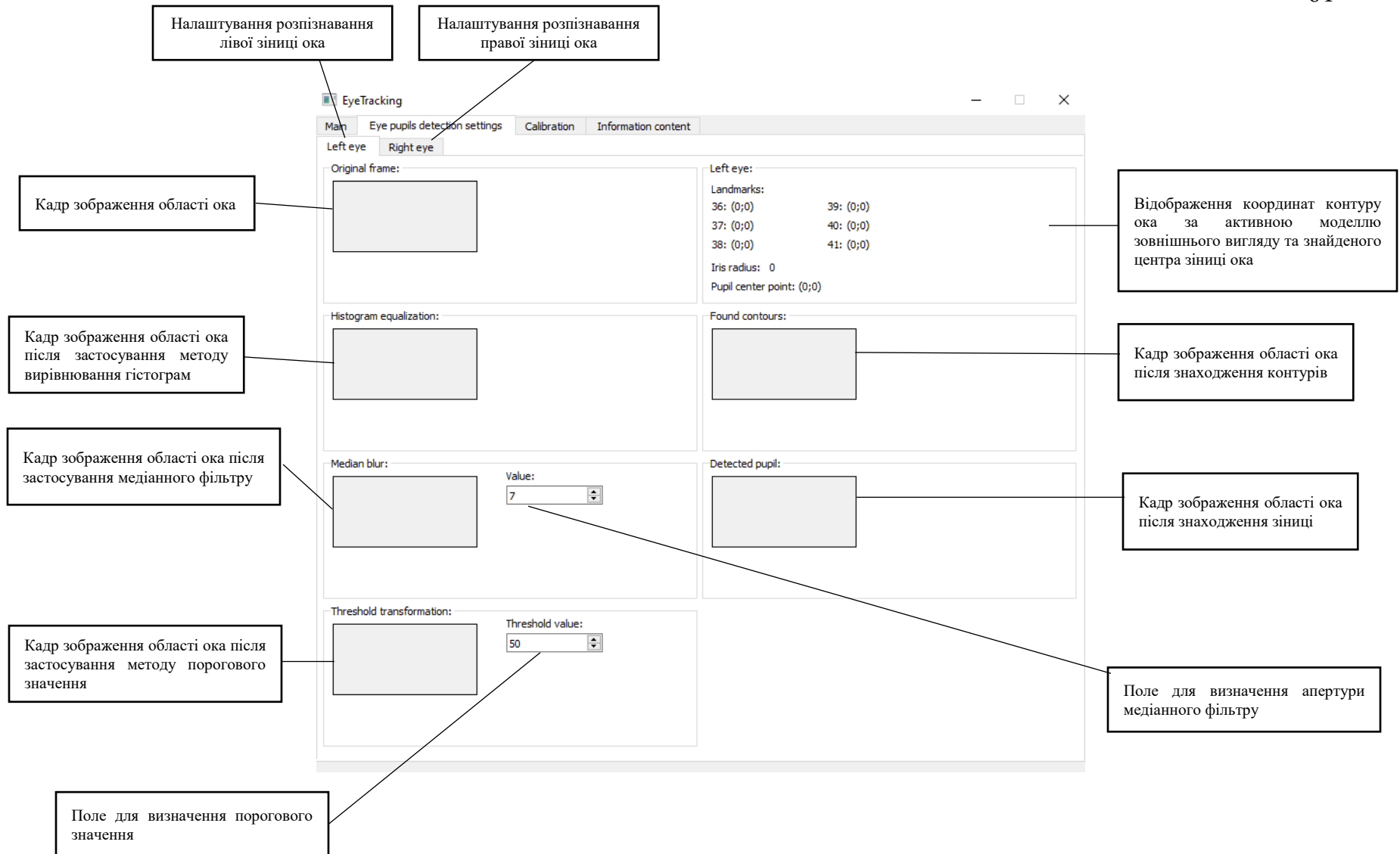


Рисунок 3.13 – Опис елементів вікна «Налаштування відстеження зіниць очей»

Вікно «Калібрування» (рис. 3.14, 3.15) дозволяє налаштовувати та проводити процес калібрування.

Для налаштування калібровки пропонується встановити наступні опції:

- форма калібрувальної точки;
- колір калібрувальної точки (у кольоровій моделі RGB);
- розмір калібрувальної точки (у пікселях);
- проміжок часу зміни позиції калібрувальної точки (у секундах);
- додаткові нотатки до опису налаштування;
- кількість позицій по горизонталі та вертикалі;
- напрямок переміщення позиції калібрувальної точки: по горизонталі, вертикалі, діагоналі;
- назва папки, де буде зберігатися результат калібрування.

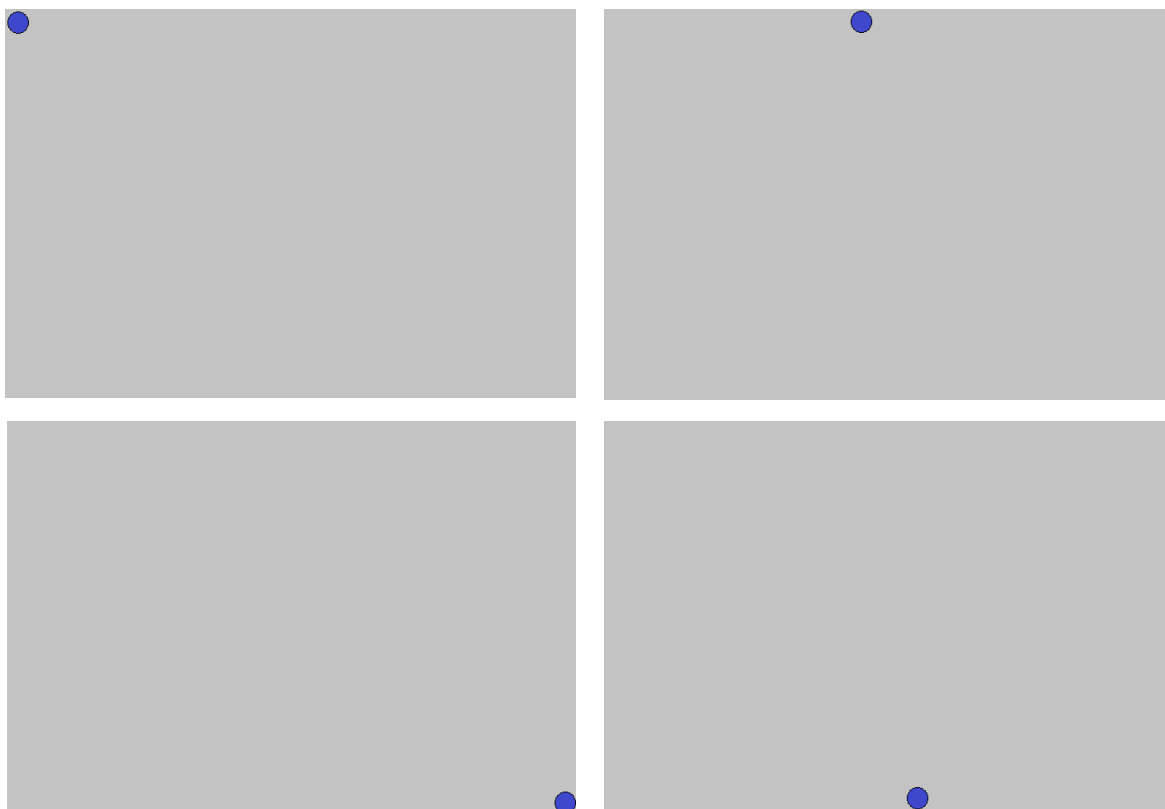


Рисунок 3.14 – Зображення вікна на повний екран при зміні позиції калібрувальної точки

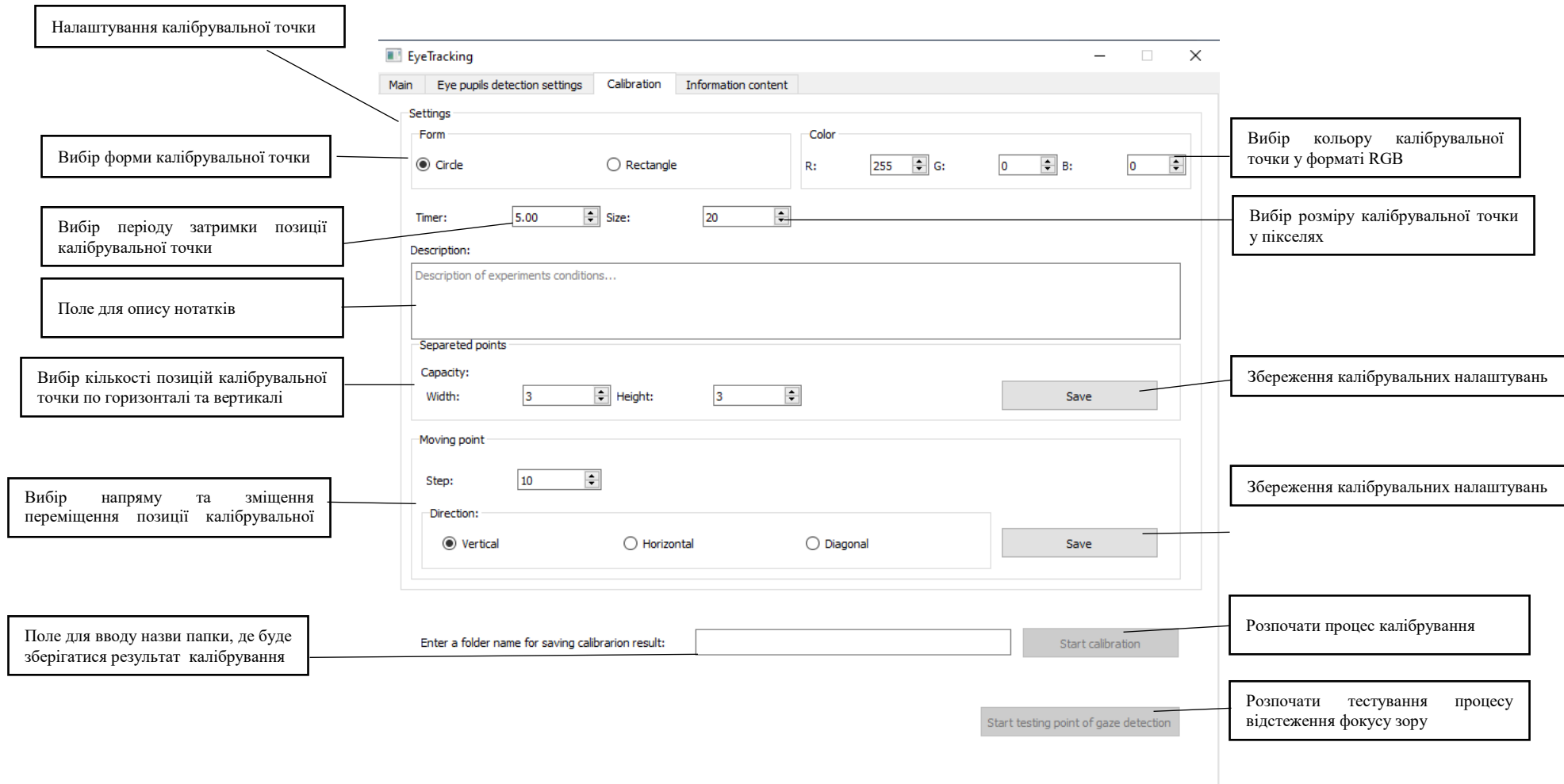


Рисунок 3.15 – Опис елементів вікна «Калібрування»

Вікно «Інформаційний контент» (рис. 3.16, 3.17) дозволяє відстежувати відповідності фокусу зору зі структурою інформації на моніторі.

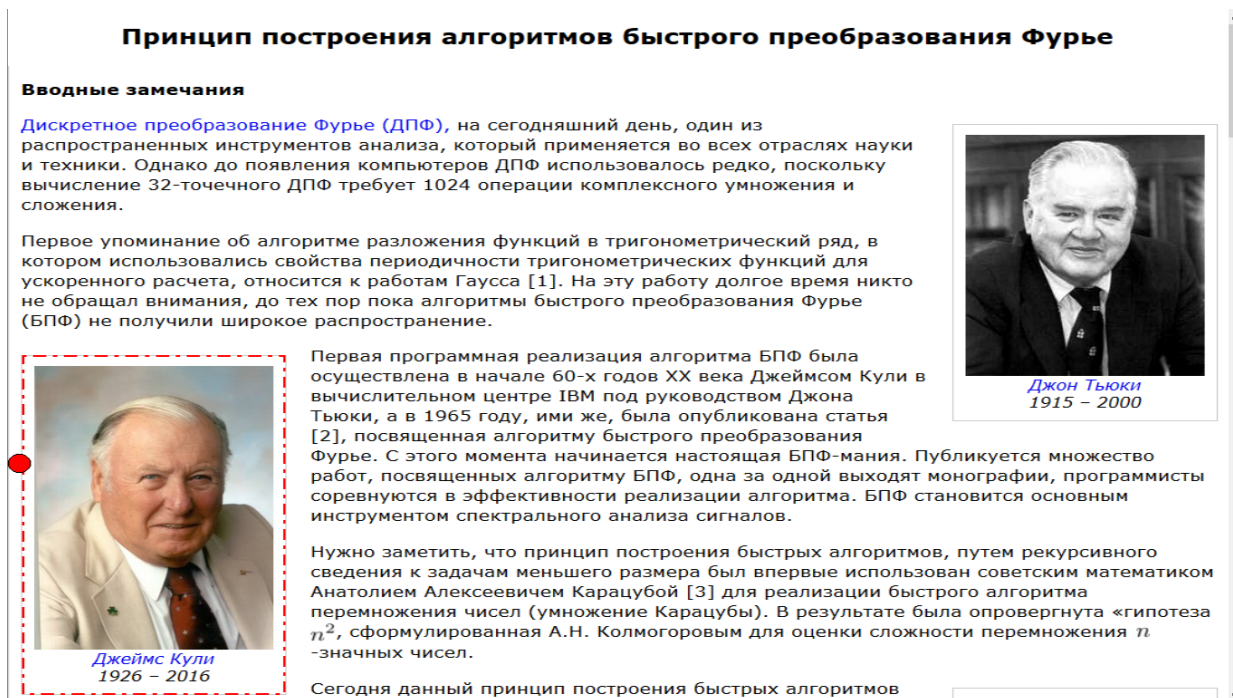


Рисунок 3.16 – Приклад відображення точки фокусу зору на інформаційному контенті

Висновки до третього розділу

Виконано проектування архітектури інструментального забезпечення для дослідження відповідності фокусу зору зі структурою інформації на моніторі. У ході проектування архітектури системи було передбачено можливість розширення системи за рахунок її розбиття на слабо пов'язані модулі. Це дозволить повторно використовувати розроблені модулі, розширювати функціонал.

У розділі наведено опис проектування інтерфейсу користувача. Для розробки інтерфейсу використовувалися знайомі користувачу елементи; увесь інтерфейс системи умовно поділено на сторінки, завдяки чому користувач з легкістю зможе знайти та змінити необхідні йому параметри та виконати дослідження.

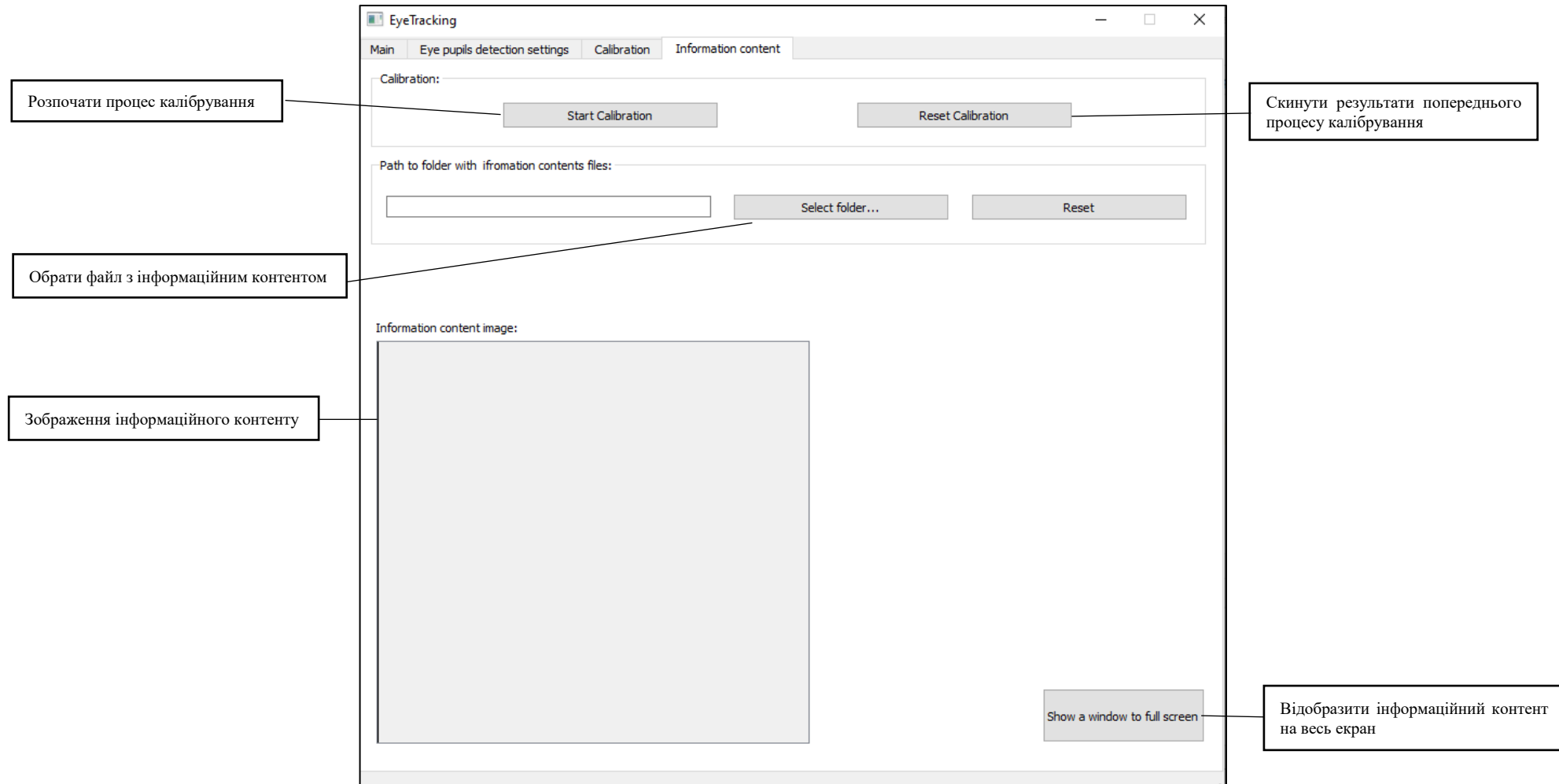


Рисунок 3.17 – Опис елементів вікна «Інформаційний контент»

4 ДОСЛІДЖЕННЯ ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ

Під час дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі було проведено декілька експериментів. Для проведення експериментів використовувалось розроблене ПЗ Eye Tracking. З апаратного середовища використовувалися наступні прилади:

- ноутбук з процесором Intel Core i5-3230M, RAM – 4Gb, відеокартою – NVIDIA GeForce GT 740M та монітором з роздільною здатністю 1366x768;
- відеокамера з роздільною здатністю 1280x720 (в даному випадку використовувалася відеокамера, вбудована в ноутбук);
- додаткова зовнішня відеокамера з роздільною здатністю 1280x1024.

Кожний експеримент проводився у рівномірно освітленому приміщенні, з використанням однієї або двох камер. Користувач знаходився навпроти ноутбуку на відстані 50 см від камери. Місце розташування камери було змінним.

4.1 Експеримент – дослідження відстеження фокусу зору користувача з використанням однієї камери при утриманні голови в статичному стані

Підготовка до експерименту

В даному експерименті використовується одна відеокамера з роздільною здатністю 1280x720, яка вбудована в ноутбук. Камера розташована на висоті 30 см від поверхні столу, в верхній частині, посередині екрану монітора.

Для дослідження відстеження фокусу зору користувача по осі абсцис та ординат використовуються різні налаштування процесу калібрування. Дані налаштування зберігаються у json форматі у файл. Приклад даних зображено на рис. 4.1.

```

*calibration_settings.json - Notepad
File Edit Format View Help
{
  "point": { "form": 0, "size": 30, "color": [255, 0, 0] },
  "timer": 15.0,
  "coordinates": [ [0, 384], [166, 384], [332, 384], [498, 384], [664, 384], [830, 384], [996, 384], [1162, 384], [1328, 384] ]
}

```

Рисунок 4.1 – Приклад даних налаштування процесу калібрування, які зберігаються у файл у json форматі

Дані налаштування процесу калібрування містять інформацію про вигляд калібрувальної точки: форма, розмір, колір та масив координат позицій її позиції.

Вісь абсцис:

- таймер зміни позиції калібрувальної точки: 15сек;
- координати позицій калібрувальної точки: [[0, 384], [166, 384], [332, 384], [498, 384], [664, 384], [830, 384], [996, 384], [1162, 384], [1328, 384]].

Вісь ординат:

- таймер зміни позиції калібрувальної точки: 15сек;
- координати позицій калібрувальної точки: [[683, 0], [683, 184], [683, 368], [683, 552], [683, 736]].

Під час процесу калібрування користувач тримає голову в статичному стані, переміщуючи тільки зіниці.

Проведення експерименту

Користувач за допомогою розробленого ПЗ процес калібрування та сліdkує за позицією калібрувальної точки. Паралельно, ПЗ відстежує переміщення зіниці ока та зберігає отримані дані у файл csv формату. Приклад вмісту файлу зображено на рис. 4.2.

R21C11															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	px	py	E36x	E36y	E41x	E41y	LPx	LPy	E42x	E42y	E47x	E47y	RPx	RPy	
2	0	384	249	217	258	221	266	216	315	219	325	220	333	215	
3	0	384	250	218	259	221	267	218	316	219	325	220	333	217	
4	0	384	249	218	258	222	268	217	315	219	325	221	333	215	
5	0	384	249	218	258	222	267	217	315	220	325	221	332	215	
6	0	384	248	219	258	222	267	218	315	220	325	221	333	216	
7	0	384	249	218	258	222	267	217	316	219	325	221	333	215	
8	0	384	248	219	258	222	267	217	315	220	325	221	333	216	
9	0	384	248	219	257	222	267	218	315	220	325	221	333	215	
10	0	384	248	219	257	222	266	218	315	220	325	221	335	217	
11	0	384	248	219	257	223	267	218	315	220	325	222	334	215	
12	0	384	248	218	257	222	267	218	314	219	324	221	334	216	
13	0	384	248	218	257	223	266	217	315	220	325	221	333	215	
14	0	384	248	219	257	223	267	218	315	220	325	222	334	216	
15	0	384	248	219	257	223	267	218	315	220	325	222	334	216	
16	0	384	248	219	257	223	267	218	315	220	325	222	333	216	
17	0	384	248	219	257	223	266	218	315	220	325	222	333	216	

Рисунок 4.2 – Вміст файлу після процесу калібрування у середовищі Excel

Файл містить декілька стовпців:

- «px», «py» – координати позиції калібрувальної точки;
- «E36x», «E36y», «E41x», «E41y» – координати контуру лівого ока (рис. 2.4.);
- «E42x», «E42y», «E47x», «E47y» – координати контуру правого ока (рис. 2.4.);
- «LPx», «LPy» – координати центру зіниці лівого ока;
- «RPx», «RPy» – координати центру зіниці правого ока.

Отже, отримавши дані виконується підрахунок координат фокусу зору та похибки різниці знайдених координат фокусу зору від координат позиції калібрувальної точки.

Результати експерименту

Підрахувавши координати фокусу зору, використовуючи отримані дані з процесу калібрування було побудовано графіки (рис. 4.3-4.4).

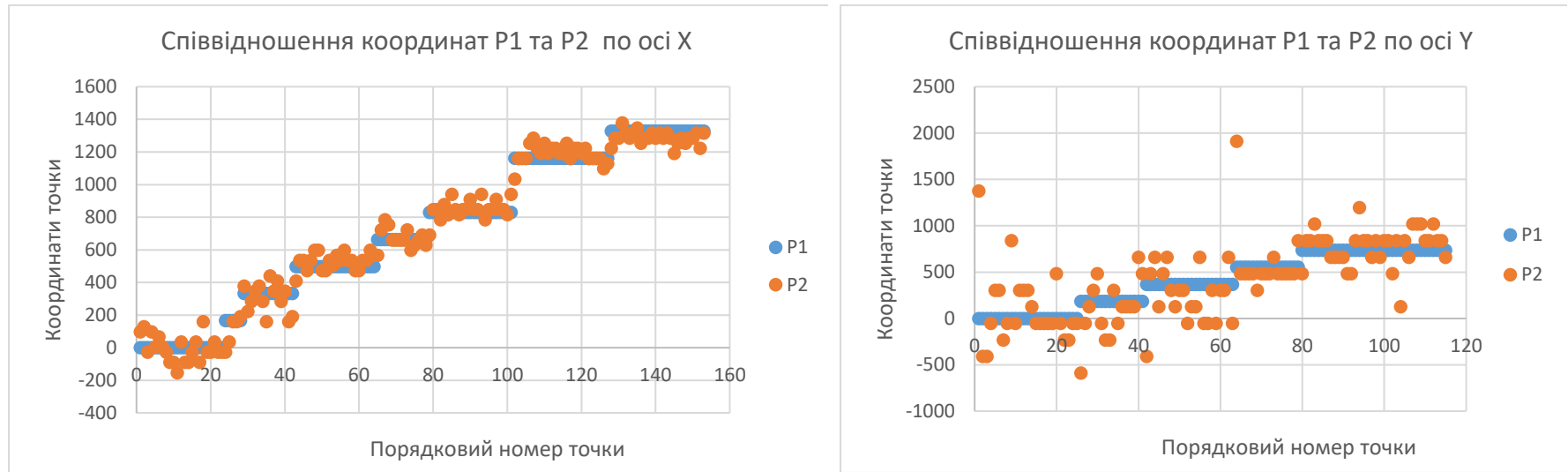


Рисунок 4.3 – Графіки співвідношення координат позицій калібрувальної точки (P1) та отриманих координат фокусу зору (P2) під час відстеження погляду користувача по обом осям

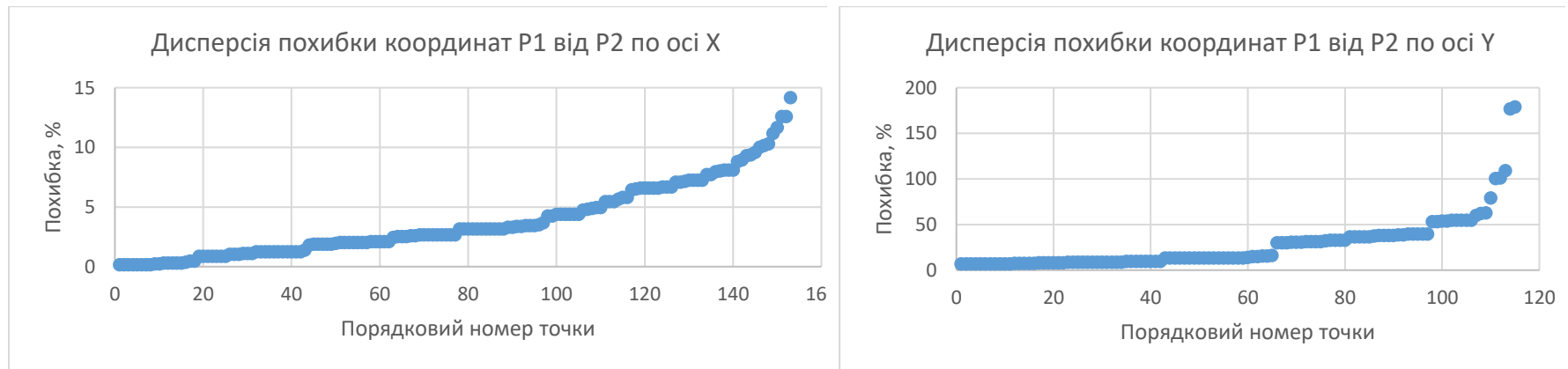


Рисунок 4.4 – Графіки дисперсії похибки координат позицій калібрувальної точки (P1) та фокусу зору (P2)

Аналізуючи графіки, можна зробити висновок, що розкид даних при отриманні координати фокусу зору по осі Y є більшим, ніж по осі X. Також це засвідчує розрахована похибка, по осі X – **3,8% (51 pixel)**, а по осі Y – **27,9% (214 pixel)**.

Інтерпретація результатів

Різниця області відстеження, по осі Y є меншою ніж по осі X, оскільки висота монітора (766px) є меншою за його ширину (1366px).

Для покращення розпізнавання фокусу погляду по осі Y було прийнято рішення про застосування додаткової відеокамери, розташованої в нижній частині посередині екрану монітора.

4.2 Експеримент – дослідження відстеження фокусу зору користувача з використанням двох камер при утриманні голови в статичному стані

Підготовка до експерименту

В експерименті використовується дві відеокамери:

- відеокамера з роздільною здатністю 1280x720, яка вбудована в ноутбук та розташована на висоті 30 см від поверхні столу, в верхній частині, посередині екрану монітора (**Камера №1**);
- додаткова зовнішня відеокамера з роздільною здатністю 1280x1024, яка розташована на висоті 10 см від поверхні столу, в нижній частині посередині екрану монітора (**Камера №2**).

Для дослідження відстеження фокусу зору користувача по осі абсцис та ординат використовуються налаштування процесу калібрування з експерименту 4.1.

Під час процесу калібрування користувач тримає голову в статичному стані, переміщуючи тільки зіниці.

Проведення експерименту

Хід проведення експерименту залишається таким же як в експерименті 4.1, тільки в даному випадку розроблене ПЗ одночасно відстежує дані з обох камер, та зберігає дані кожної відеокамери в окремий файл. Підрахувавши координати фокусу зору та похибку для кожної камери, розраховується їх середнє значення.

Результати експерименту

Результати експерименту представлені на графіках рис. 4.5-4.10 та в таблиці 4.1.

Таблиця 4.1 – Похибки різниці знайдених координат фокусу зору від координат позиції калібрувальної точки під час використання двох відеокамер

	По осі X		По осі Y	
	Похибка у пікселях (px)	Похибка у відсотках (%)	Похибка у пікселях (px)	Похибка у відсотках (%)
Камера №1	90	6,6	90	11.7
Камера №2	67	4,9	87	11.4
Обидві камери	58	4,2	71	9.3

Інтерпретація результатів

Використання обох камер одночасно дають кращі результати ніж використання їх окремо. Також, введені зміни покращили ситуацію розпізнавання координат фокусу зору, так як похибка по осі Y становить **9,3% (71 pixel)**, а це майже в 3 рази менше ніж у попередньому експерименті. Проте похибка по осі X незначно збільшилась та становить **4,2% (58 pixel)**.

4.3 Експеримент – дослідження відстеження фокусу зору користувача з використанням зміни позиції калібрувальної точки по обом осям

Підготовка до експерименту

В експерименті використовується дві відеокамери, які описані в експерименті №2.

Для дослідження відстеження фокусу зору користувача використовуються наступні налаштування процесу калібрування:

- таймер зміни позиції калібрувальної точки: 10сек;
- координати позицій калібрувальної точки: [[0, 0], [683, 0], [1346, 0], [0, 384], [683, 384], [1346, 384], [0, 748], [683, 748], [1346, 748]].

Під час процесу калібрування користувач тримає голову в статичному стані, переміщуючи тільки зіниці.

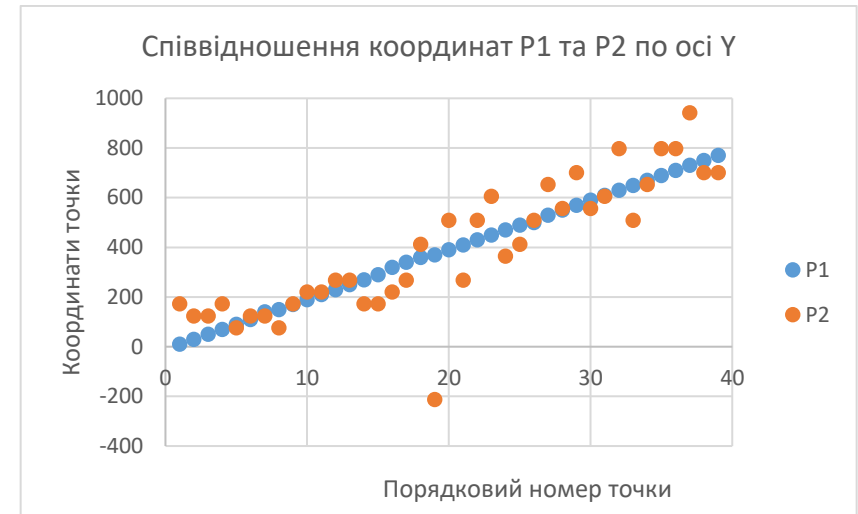
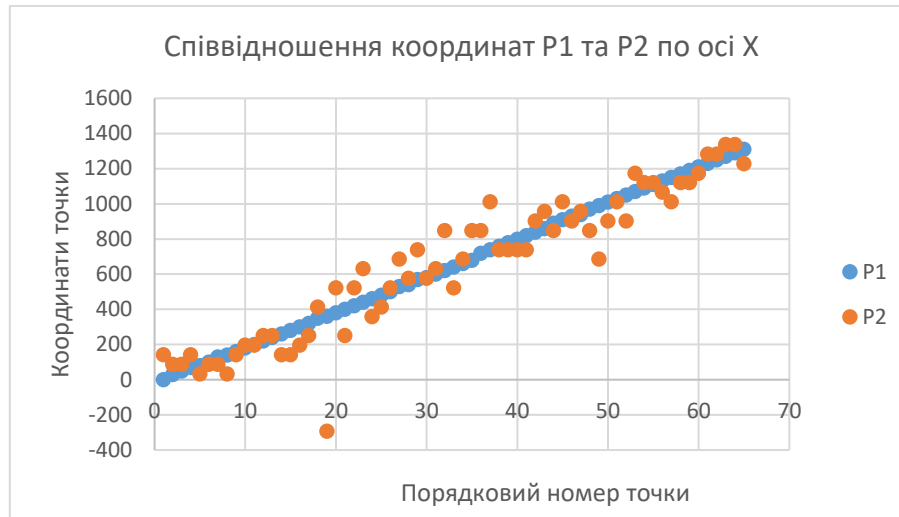


Рисунок 4.5 – Графіки співвідношення координат позицій калібрувальної точки (P1) та отриманих координат фокусу зору (P2) під час відстеження погляду користувача для камери №1

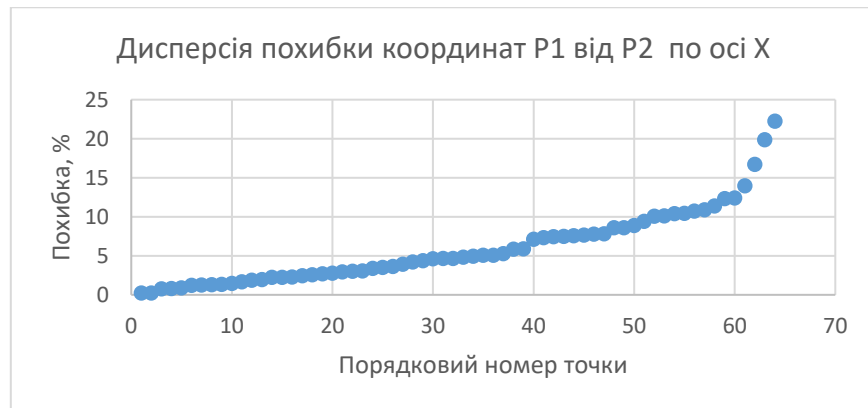


Рисунок 4.6 – Графіки дисперсії похибки координат позицій калібрувальної точки (P1) та фокусу зору (P2) для камери №1

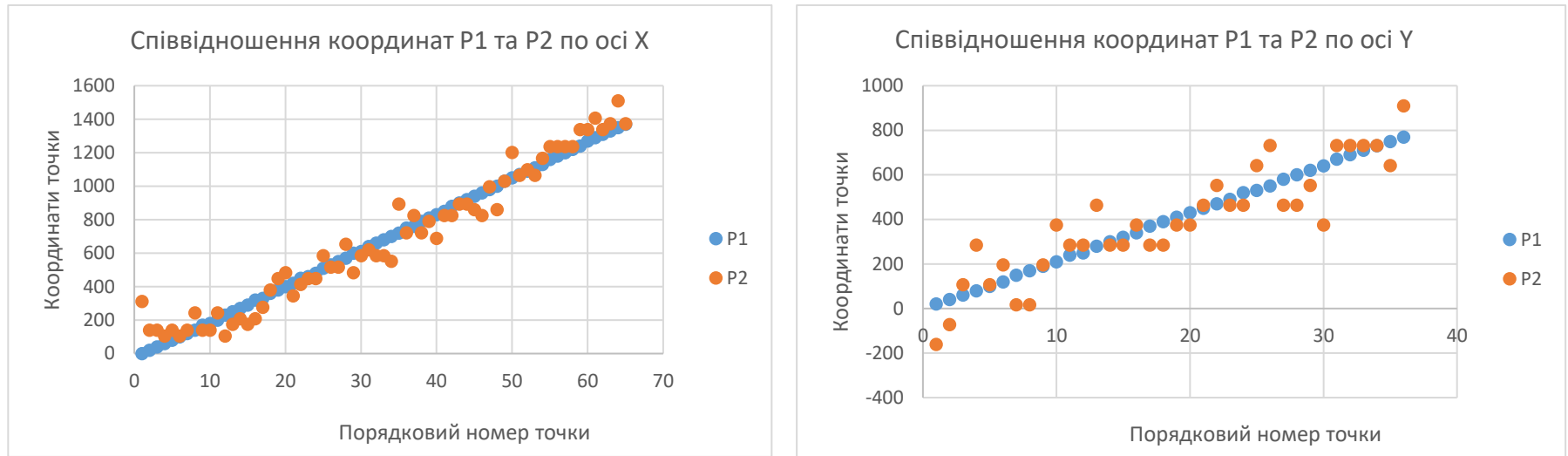


Рисунок 4.7 – Графіки співвідношення координат позицій калібрувальної точки (P1) та отриманих координат фокусу зору (P2) під час відстеження погляду користувача для камери №2

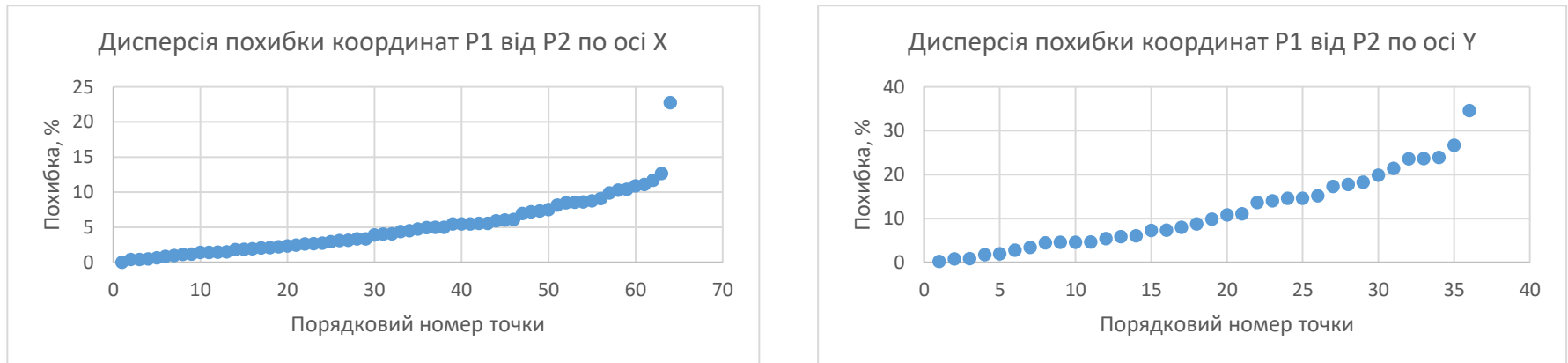


Рисунок 4.8 – Графіки дисперсії похибки координат позицій калібрувальної точки (P1) та фокусу зору (P2) для камери №2

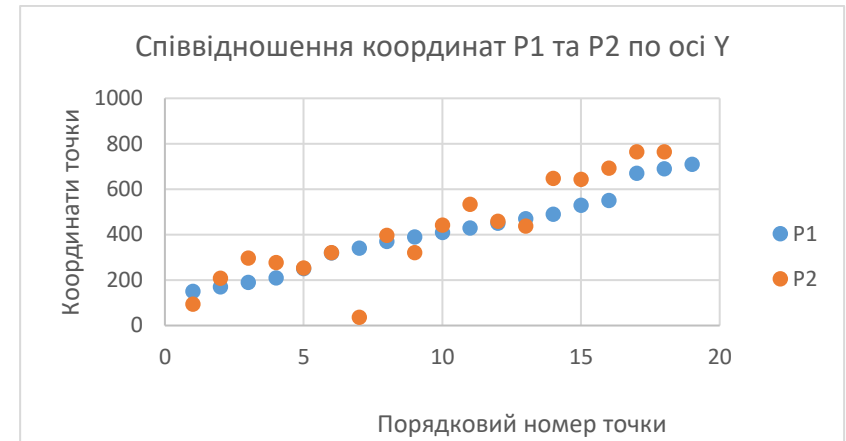
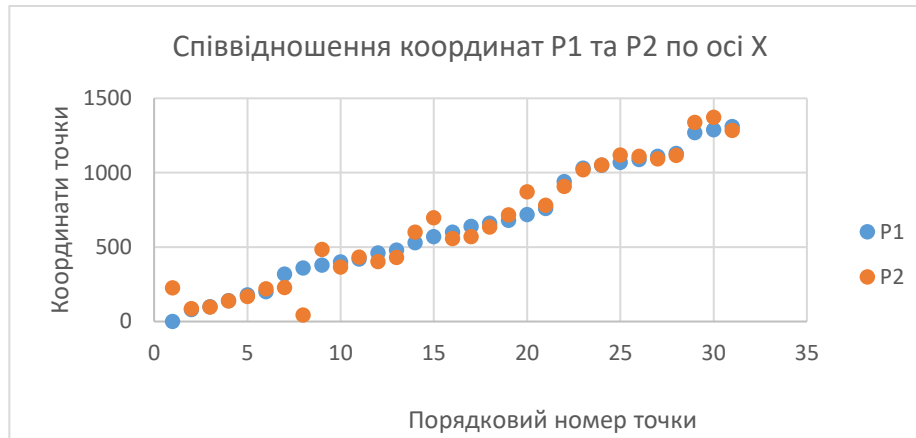


Рисунок 4.9 – Графіки співвідношення координат позицій калібрувальної точки (P1) та отриманих координат фокусу зору (P2) під час відстеження погляду користувача для обох камер

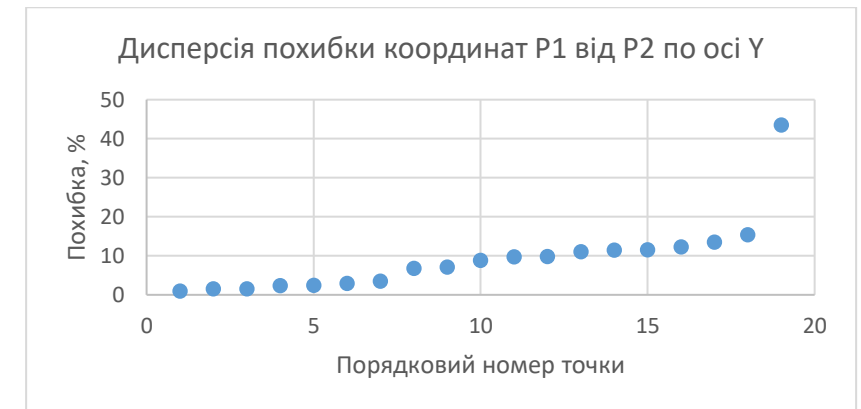
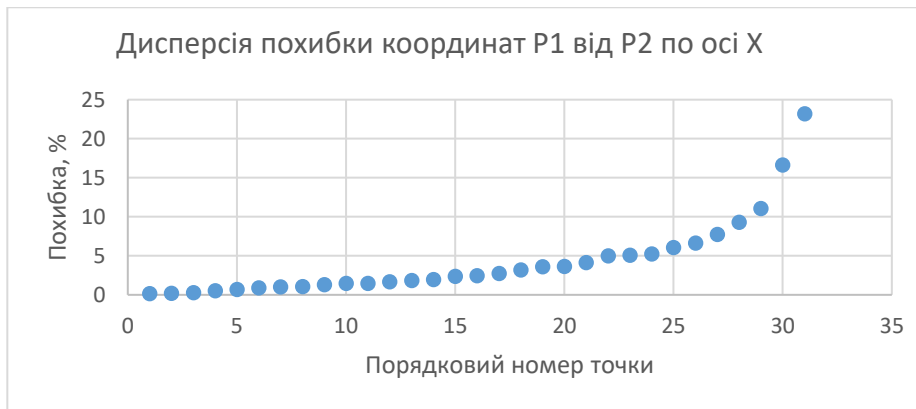


Рисунок 4.10 – Графіки дисперсії похибки координат позицій калібрувальної точки (P1) та фокусу зору (P2) для обох камер

Проведення експерименту

Хід проведення експерименту залишається таким же як в експерименті 4.2.

Результати експерименту

Результати експерименту представлені на графіках рис. 4.11-4.12 та в таблиці 4.2.

Таблиця 4.2 – Похибки різниці знайдених координат фокусу зору від координат позиції калібрувальної точки для двох відеокамер

По осі X		По осі Y	
Похибка у пікселях (px)	Похибка у відсотках (%)	Похибка у пікселях (px)	Похибка у відсотках (%)
84	6.2	94	12.3

Інтерпретація результатів

Результат відстеження координат фокусу зору погіршився по обом осям в порівнянні з експериментом 4.2:

- по осі X: був **4,2% (58 pixel)**, став **6,2% (84 pixel)**;
- по осі Y: був **9,3% (71 pixel)**, став **12,3% (94 pixel)**.

В даному експерименті, використовувалось 3 зміни позиції калібрувальної точки під час процесу калібрування по обом осям, на відміну від експерименту №2, де по осі X використовувалось 9 позицій, по осі Y – 5. Кількість змін позицій калібрувальної точки вплинуло на похибку відстеження фокусу зору.

Результат експерименту показав, що використання зміни 3 позицій калібрувальної точки не є достатньо. Це свідчить проте, що необхідно використовувати більше позицій калібрувальної точки під час процесу калібрування, щоб краще підібрати параметри для підрахунку координат фокусу зору.

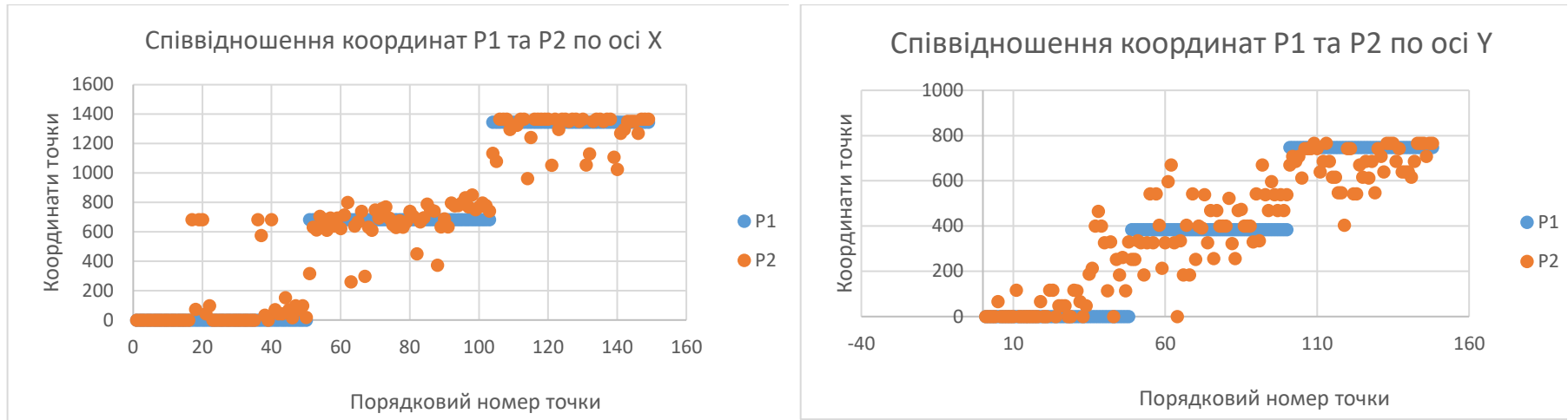


Рисунок 4.11 – Графіки співвідношення координат позицій калібрувальної точки (P1) та отриманих координат фокусу зору (P2) під час відстеження погляду користувача для обох камер

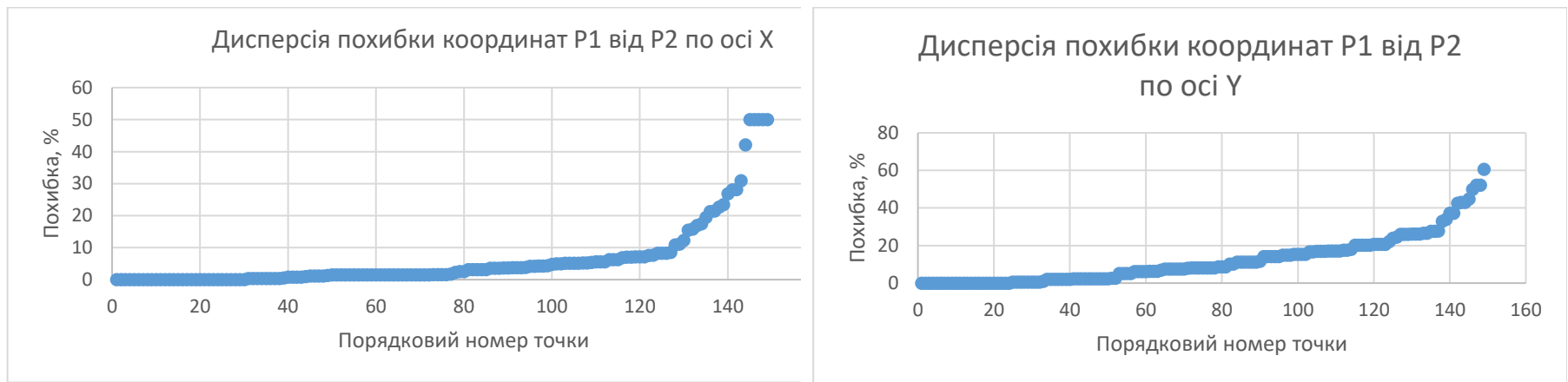


Рисунок 4.12 – Графіки дисперсії похибки координат позицій калібрувальної точки (P1) та фокусу зору (P2) для обох камер

4.4 Експеримент – дослідження відстеження фокусу зору користувача з використанням збільшеної кількості позицій калібрувальної точки

Підготовка до експерименту

В експерименті використовується дві відеокамери, які описані в експерименті 4.2.

Для дослідження відстеження фокусу зору користувача використовуються наступні налаштування процесу калібрування:

- таймер зміни позиції калібрувальної точки: 6 с;
- координати позицій калібрувальної точки: [[0, 0], [273, 0], [546, 0], [819, 0], [1092, 0], [1366, 0], [0, 153], [273, 153], [546, 153], [819, 153], [1092, 153], [1366, 153], [0, 307], [273, 307], [546, 307], [819, 307], [1092, 307], [1366, 307], [0, 460], [273, 460], [546, 460], [819, 460], [1092, 460], [1366, 460], [0, 614], [273, 614], [546, 614], [819, 614], [1092, 614], [1366, 614], [0, 768], [273, 768], [546, 768], [819, 768], [1092, 768], [1366, 768]].

Під час процесу калібрування користувач тримає голову в статичному стані, переміщуючи тільки зіниці.

Проведення експерименту

Хід проведення експерименту залишається таким же як в експерименті 4.2. На рис. 4.13 зображено монітор екрану, під час проведення експерименту.

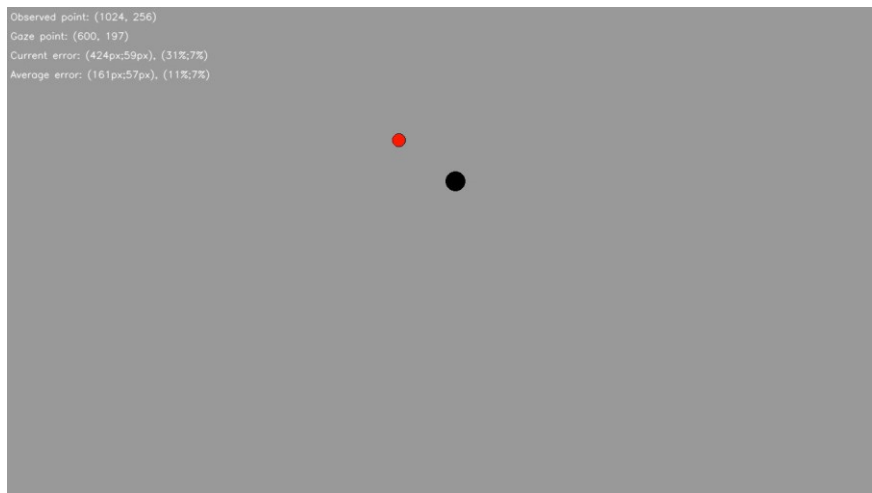


Рисунок 4.13 – Монітор екрану під час експерименту

На екрані зображено дві точки: чорного кольору – позиція калібрувальної точки та червоного кольору – позиція фокусу зору користувача. В лівому верхньому кутку екрану відображено координати позиції калібрувальної точки та фокусу зору, а також значення поточної та загальної похибки.

Результати експерименту

Результати експерименту наведені на графіках рис. 4.14-4.15 та в таблиці 4.3.

Таблиця 4.3 – Похибки різниці знайдених координат фокусу зору від координат позиції калібрувальної точки для двох відеокамер

По осі X		По осі Y	
Похибка у пікселях (px)	Похибка у відсотках (%)	Похибка у пікселях (px)	Похибка у відсотках (%)
80	5,8	73	9

Інтерпретація результатів

Результат відстеження координат фокусу зору трохи покращився, завдяки збільшенню позицій калібрувальної точки під час процесу калібрування, в порівнянні з експериментом 4.3.

4.5 Експеримент – дослідження відстеження фокусу зору користувача з утриманням голови користувача в динамічному стані.

Підготовка до експерименту

В експерименті використовується дві відеокамери, які описані в експерименті 4.2.

Для дослідження відстеження фокусу зору користувача використовуються налаштування процесу калібрування з експерименту 4.4.

Під час процесу калібрування користувач тримає голову в динамічному стані, повертаючи голову під час експерименту.

Проведення експерименту

Хід проведення експерименту залишається таким же як в експерименті 4.4.

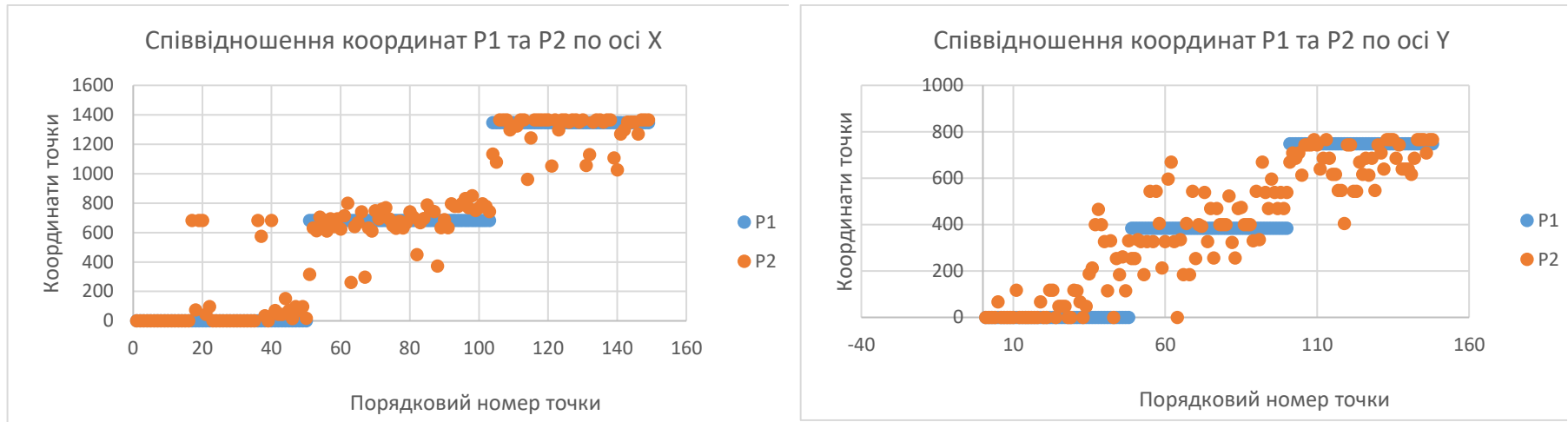


Рисунок 4.14 – Графіки співвідношення координат позицій калібрувальної точки (P1) та отриманих координат фокусу зору (P2) під час відстеження погляду користувача для обох камер

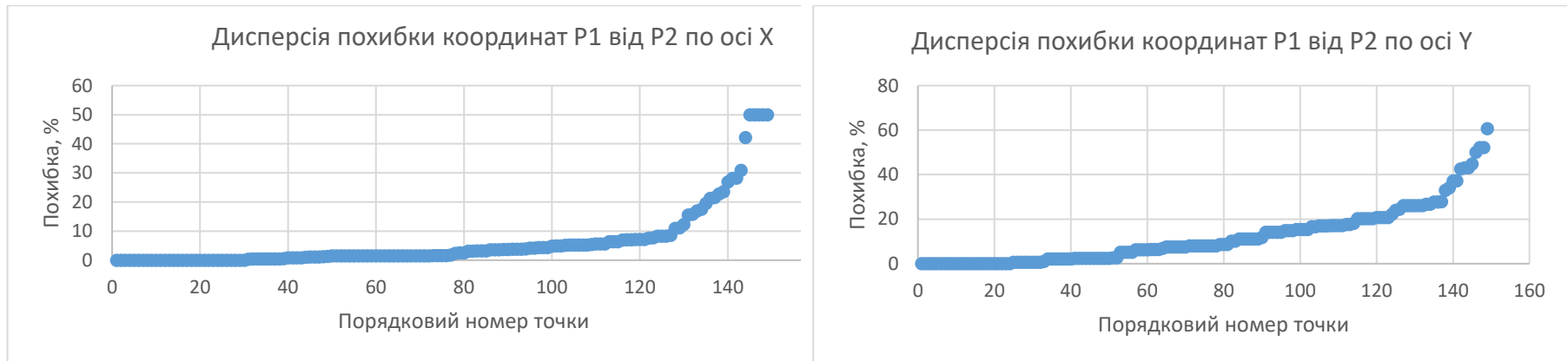


Рисунок 4.15 – Графіки дисперсії похибки координат позицій калібрувальної точки (P1) та фокусу зору (P2) для обох камер

Результати експерименту

Результати експерименту наведені в таблиці 4.4.

Таблиця 4.4 – Похибки різниці знайдених координат фокусу зору від координат позиції калібрувальної точки для двох відеокамер

По осі X		По осі Y	
Похибка у пікселях (px)	Похибка у відсотках (%)	Похибка у пікселях (px)	Похибка у відсотках (%)
223	16	127	16

Інтерпретація результатів

Результат відстеження координат фокусу зору під час динамічного руху голови є гіршим ніж коли голова знаходиться в статичному стані (порівнюючи з експериментом №4). Це є очікуваний результат, так як під час руху голови складніше розпізнати рух зіниці ока.

4.6 Експеримент – дослідження відстеження фокусу зору користувача з утриманням голови користувача в динамічному стані та зміною розташування додаткової зовнішньої відеокамери

Підготовка до експерименту

В експерименті використовується дві відеокамери:

- відеокамера з роздільною здатністю 1280x720, яка вбудована в ноутбук та розташована на висоті 30 см від поверхні столу, в верхній частині, посередині екрану монітора;
- додаткова зовнішня відеокамера з роздільною здатністю 1280x1024, яка розташована на висоті 25 см від поверхні стола, посередині екрану монітора.

Для дослідження відстеження фокусу зору користувача використовуються налаштування процесу калібрування з експерименту № 4.4.

Під час процесу калібрування користувач тримає голову в динамічному стані, повертаючи голову під час експерименту.

Проведення експерименту

Хід проведення експерименту залишається таким же як в експерименті №2.

Результати експерименту

Результати експерименту наведені в таблиці 4.5.

Таблиця 4.5 – Похибки різниці знайдених координат фокусу зору від координат позиції калібрувальної точки для двох відеокамер

По осі X		По осі Y	
Похибка у пікселях (px)	Похибка у відсотках (%)	Похибка у пікселях (px)	Похибка у відсотках (%)
193	14	109	14

Інтерпретація результатів

Похибка відстеження координат фокусу зору незначно зменшилась за рахунок зміни положення додаткової відеокамери, порівнюючи з експериментом №5. Незважаючи на покращення результату відстеження, зазначене місце розташування додаткової камери не є зручним, тому що відеокамера заважає користувачу спостерігати за екраном монітора.

Порівняльні результати експериментів наведені у таблиці 4.6.

Таблиця 4.6 – Узагальнені результати похибки відстеження фокусу зору проведених експериментів по осям

Експеримент	По осі X		По осі Y	
	Похибка у пікселях (px)	Похибка у відсотках (%)	Похибка у пікселях (px)	Похибка у відсотках (%)
Експеримент 4.1	51	3,8	214	27,9
Експеримент 4.2	58	4,2	71	9.3
Експеримент 4.3	84	6.2	94	12.3
Експеримент 4.4	80	5,8	73	9
Експеримент 4.5	223	16	127	16
Експеримент 4.6	193	14	109	14

Висновки до четвертого розділу

Для дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі було проведено декілька експериментів за різних умов: використання однієї або двох камер, використання різних налаштувань процесу

калібрування, зміна положення додаткової відеокамери, стан голови користувача під час експерименту.

Результати експериментів показали, що використання двох відеокамер дало кращий результат відстеження фокусу зору по осі Y. Також, було встановлено, що налаштування процесу калібрування впливає на результат відстеження фокусу зору.

Порівнюючи результати експериментів, де користувач тримає голову в статичному або динамічному стані, було встановлено, що результат відстеження фокусу зору є кращим під час утримання голови в статичному стані, тому що система відстеження краще розпізнає переміщення зіниць за цих умов.

На підставі проведених експериментів дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі було встановлено, що похибка розпізнавання фокусу зору становить 14% по обом осям.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Вимоги безпеки при виконанні робіт на робочому місці

Заходи з охорони праці є важливим моментом задля вдосконалення умов праці та підвищення високопродуктивної і творчої роботи. Цим питанням займаються всі роботодавці, адже турбота про здоров'я людини це елемент конкуренції роботодавців в питанні залучення кадрів. Для правильної організації процесу трудової діяльності людини необхідні знання в області фізіології праці, бо тільки сприятливі умови праці сприяють розвитку здібностей, зниженню втомлюваності та сприяє зниженню аварійності.

В даній магістерській роботі було створено програму з відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК, яка супроводжується виконанням ряду робіт з використанням ЕОМ. Тому необхідним є дотримання правил охорони праці, техніки безпеки та протипожежної безпеки при роботі з комп'ютерною технікою.

Сприятливі та безпечні умови праці програміста-розробника забезпечуються наступними нормативними документами:

- ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», затвердженими Постановою головного державного санітарного лікаря України 10 грудня 1998 р. № 7 [30];
- ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки, затверджено та введено в дію наказом міністерства економічного розвитку і торгівлі України 14.10.2013 № 1231[31];
- ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення [32];

- ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища, введено в дію з 01.01.2006 [33];
- НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», затвердженими Наказом Міністерства соціальної політики України 14.02.2018 № 207 [34];
- ДБН В.2.5-28:2018 Державні будівельні норми України «Природне і штучне освітлення», затверджені наказом Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 03.10.2018 № 264, введено в дію з 01.03.2019 [35];
- ДСН 3.3.6.042-99 «Державні санітарні норми мікроклімату виробничих приміщень», затверджені Постановою головного санітарного лікаря України № 42 від 1 грудня 1999 року [36];
- ДСН 3.3.6.037-99 «Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку», затверджені Постановою головного санітарного лікаря України № 37 від 1 грудня 1999 року [37];
- Закон України «Про охорону праці» зі змінами і доповненнями, введений в дію Постановою ВР № 2695-XII від 14.10.92, ВВР, 1992, № 49, ст.669 [38];
- «Керівництво з визначення і оцінки важкості та напруженості трудового процесу» НАМН України від 02.05.2019р [39];
- ДСанПіН «Гігієнічна класифікація праці», затверджені Наказом Міністерства охорони здоров'я України №248 від 08.04.14р [40];
- НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні», затверджені Наказом Міністерства внутрішніх справ України 30 грудня 2014 року N 1417, зареєстровані в Міністерстві юстиції України 05 березня 2015 р. за N 252/26697 [41];
- Кодекс цивільного захисту України зі змінами та доповненнями, Відомості Верховної Ради (ВВР), 2013, № 34-35, ст.458 [42].

5.2 Шкідливі виробничі фактори на робочому місці

Під час роботи на програміста можуть впливати наступні небезпечні та шкідливі виробничі фактори: невідповідність мікроклімату у приміщенні, недостатнє освітлення, підвищений рівень шуму та вібрації, електромагнітні випромінювання, висока напруга електричної мережі та інші.

Значним небезпечним фактором під час тривалої роботи з ПК є втомлюваність зорового апарату, що спричинена високим ступенем напруженості трудового процесу.

Неправильна організація робочого місця призводить до втомлюваності, м'язовій напрузі та розвитку захворювань з боку опорно-рухового апарату.

У даному розділі розглядаються умови в приміщенні, де проводились дослідження відповідності фокусу зору зі структурою інформації на моніторі ПК.

5.2.1 Організація робочого місця

Приміщення, де проводились дослідження, має загальну площу $17,5 \text{ м}^2$, а об'єм $52,5 \text{ м}^3$, що повністю відповідає вимогам [30], де зазначено, що площа одного робочого місця оператора ПК не повинна бути меншою за 6 м^2 , а об'єм не менший за 20 м^3 . У приміщенні розташоване одне робоче місце.

Робоче місце оператора та взаємне розташування елементів робочого місця обладнано згідно з [31] та [32].

Комп'ютеризоване робоче місце розміщено на відстані $1,2 \text{ м}$ від стіни зі світловими прорізами, головний потік світла прямує зліва.

Важливим моментом є розміри меблів на комп'ютеризованих робочих місцях, зокрема робочого столу. Робочий стіл має достатній розмір для зручного користування та має поверхню з низькою відбивною здатністю. Робоче крісло досить стійке і дозволяє легко пересуватися та займати зручне положення. Сидіння регулюється по висоті, спинка сидіння – як по висоті, так і по нахилу. Для зручності передбачено підніжку. Робоче місце програміста повністю відповідає ергономічним вимогам [32], [33] та [34].

5.2.2 Вимоги до освітлення у приміщенні

В приміщенні використовується природне та штучне освітлення. Природне освітлення здійснено через світлові прорізи, які забезпечують коефіцієнт природної освітленості (КПО) не нижче 1,5%. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури на вікнах встановлені жалюзі. Штучне освітлення в приміщенні здійснено системою загального рівномірного освітлення.

Характеристика зорової роботи програміста відноситься до робіт високої точності (III в). Згідно з [35] нормативне значення загального освітлення робочого приміщення повинно бути $E = 200-300 \text{ Лк}$.

В приміщенні, де проводились дослідження виміряли освітлення за допомогою люксметра Ю-117 (зав. номер 2331). Рівень природної освітленості поверхні, де розташований ПК програміста, складає 320 Лк, що цілком задовольняє встановленим у [35] вимогам.

5.2.3 Вимоги до мікроклімату приміщення

Показники мікроклімату приміщення, який визначається температурою, вологістю, швидкістю руху повітря, а також температурою внутрішніх поверхонь приміщення (стін, стелі, підлоги, технічного обладнання) та впливає на теплообмін людини з навколишнім середовищем, її тепловий стан, самопочуття, працездатність і здоров'я відповідають встановленим санітарно-гігієнічним вимогам [36].

Робота програміста виконується сидячи і не потребує фізичного напруження, тому її можна віднести до категорії Ia. Відповідно до [36] категорія Ia – це легкі фізичні роботи, при яких витрата енергії дорівнює 105-140 Вт (90-120 ккал/год.). Тому встановлені наступні оптимальні значення параметрів мікроклімату:

- холодний період року:
 - температура повітря 22-24⁰С;
 - відносна вологість 60-40 %;
 - швидкість руху повітря 0,1 м/с;

- теплий період року:
 - температура повітря 23-25⁰С;
 - відносна вологість 60-40 %;
 - швидкість руху повітря 0,1 м/с;

Фактичні значення температури, відносної вологості та швидкості руху повітря наведено в таблиці 5.1.

Таблиця 5.1 – Порівняння фактичних величин температури, відносної вологості та швидкості руху повітря з нормованими

Параметр, що визначається	Засіб вимірювальної техніки	Виміряне значення	Нормоване значення [36]	Висновок
Температура повітря, ⁰ С	Психрометр аспіраційний, МВ-4М, зав. № 24555	23	22-24	відповідає
Відносна вологість, %	Психрометр аспіраційний, МВ-4М, зав. № 24555	52	60-40	відповідає
Швидкість руху, м/с	Вимірювач швидкості газових потоків ІС-2, зав. № 66	<0.1	0,1	відповідає

5.2.4 Вимоги до шуму та вібрації у приміщенні

Рівні звукового тиску в октавних смугах частот, рівні звуку, еквівалентні рівні звуку та вібрація на робочих місцях у приміщення нормуються згідно [30] та [37].

Зазвичай, робоче місце програміста складається з ПК, кожний з яких устаткований монітором, вінчестером в системному блоці, трьома вентиляторами системи охолодження ПК та клавіатурою. Крім того поряд працює периферійна техніка. Таким чином у приміщенні мають місце шуми механічного і аеродинамічного походження, широкосмугові із аперіодичним підсиленням при роботі принтерів. Орієнтовні еквівалентні рівні звукового тиску джерел шуму, що діють на програміста на його робочому місці, представлені в таблиці 5.2.

Таблиця 5.2 – Рівні звукового тиску від різних джерел

Джерело шуму	Рівень шуму, дБА
Жорсткий диск стаціонарного комп'ютера	45
Вентилятор на стаціонарному комп'ютері	45
Принтер матричний	55
Сканер	50

Дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК проводилось з використанням ноутбуку, що має значні переваги перед стаціонарними комп'ютерами, оскільки рівень шуму від ноутбуку значно менший. У стаціонарних комп'ютерах часто набридливо шумлять вентилятори, оскільки вони громіздкі і єдиний варіант зниження шуму – заміна вентилятора на більш якісний.

У ноутбуках така проблема зустрічається лише у випадку перегріву з якоїсь причини.

Відповідно до [37] допустимий еквівалентний рівень шуму для робочого місця програміста складає 50 дБА, що перевищує рівень шуму від ноутбуку.

5.2.5 Вимоги до напруженості праці

Робота програміста-розробника пов'язана з інтенсивною розумовою творчою працею і тривалим зосередженням уваги на екран монітора, що призводить до втомлюваності зорового апарату.

Тому саме очі найбільше страждають під час роботи з комп'ютером. Велике значення при роботі з комп'ютером мають такі речі як відстань до екрана, шрифт, розмір тексту на моніторі, наявність або відсутність мерехтіння, яскравість екрана, освітлення робочого місця, наявність перерв у роботі.

Відповідно до Закону України «Про охорону праці» [38] необхідно передбачати додаткові нетривалі перерви в періоди, що передують появі ознак стомлення і зниження працездатності.

Перерви для відпочинку та тривалість робочої зміни під час створення програми з відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК

встановлюють згідно з [30, 39, 40]. Для розробників програм слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за персональним комп'ютером.

Під час проведення експериментів при дослідженні відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК рекомендована відстань до екрану монітора, яка складає 50 см, що є мінімально допустимою відстанню.

5.3 Дії працівників в надзвичайних ситуаціях

До надзвичайних ситуацій техногенного характеру при роботі з ПК належать пожежі.

Правила пожежної безпеки в Україні [41] та Кодекс цивільного захисту України [42] визначають загальні правові, економічні та соціальні основи забезпечення пожежної безпеки на території України, регулюють відносини державних органів, юридичних і фізичних осіб у цій галузі незалежно від виду їх діяльності та форм власності.

Пожежна безпека – стан об'єкта, при якому з регламентованою ймовірністю виключається можливість виникнення та розвиток пожежі і впливу на людей її небезпечних факторів, а також забезпечується захист матеріальних цінностей. Для забезпечення пожежної безпеки в установах проводять пожежну профілактику, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі.

Причинами виникнення пожеж можуть бути:

- несправності електропроводки, розеток і вимикачів які можуть привести до короткого замикання або пробією ізоляції;
- використання ушкоджених (несправних) електроприладів;
- використання в приміщенні електронагрівальних приладів з відкритими нагрівальними елементами;
- виникнення пожежі внаслідок влучення блискавки в будинок;
- загоряння будинку внаслідок зовнішніх впливів;

– неакуратне поводження з вогнем і недотримання мір пожежної безпеки.

У випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону пожежну команду, евакуювати людей із приміщення і приступити до ліквідації пожежі первинними засобами пожежогасіння такими як: вогнегасники, пожежний інвентар (покривала з негорючого теплоізоляційного полотна, ящики з піском, пожежні відра, совкові лопати, ломи, сокири тощо), системи автоматичного пожежогасіння.

При наявності невеликого вогнища полум'я, можна скористатися підручними засобами з метою припинення доступу повітря до об'єкта загоряння.

Висновки до п'ятого розділу

При дослідженні факторів виробничого середовища і трудового процесу [40] на робочому місці програміста-розробника було виявлено можливі небезпечні та шкідливі виробничі фактори: невідповідність мікроклімату у приміщенні, недостатнє освітлення, підвищений рівень шуму та вібрації, електромагнітні випромінювання, висока напруга електричної мережі та інші.

Значним небезпечним фактором під час тривалої роботи з ПК є втомлюваність зорового апарату, що спричинена високим ступенем напруженості трудового процесу.

Яскраве світло в районі периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності. Тому освітлення в приміщенні повинно бути досить рівномірним, ступінь освітлення приміщення і яскравість екрану повинні бути приблизно однаковими.

Для зниження нервово-емоційного напруження і втоми очей доцільно деякі перерви використовувати для комплексу вправ, дотримуватись безпечної відстані від очей до екрану монітора та слідкувати за правильним положенням тулуба під час роботи.

При створенні програми з відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК було обрано таке устаткування, яке не створює зайвого шуму та не виділяє надлишкового тепла.

Дослідження проводилось з використанням ноутбуку з процесором Intel Core i5-3230M, RAM – 4Gb, відеокартою – NVIDIA GeForce GT 740M та монітором з роздільною здатністю 1366x768, що значно зменшує рівень шуму порівняно з використанням стаціонарних комп'ютерів.

Вплив небезпечних та шкідливих факторів на робочому місці програміста-розробника необхідно мінімізувати задля зменшення ризиків погіршення стану його здоров'я. Це реалізується дотриманням законодавчої бази з охорони праці, яка в нашій країні з кожним роком удосконалюється, бо пріоритетним залишається збереження життя і здоров'я працюючих.

ЗАГАЛЬНІ ВИСНОВКИ

У роботі було виконано дослідження розробки засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера з використанням апаратних засобів, які доступні кожному для покращення засвоєння інформації студентами під час дистанційного навчання.

Реалізовано програмне середовище, яке здатне дослідити відстеження відповідності фокусу зору зі структурою інформації на моніторі з використанням однієї або двох відеокамер.

У ході розробки проекту використовувалися новітні технології. Завдяки дотриманню принципів об'єктно-орієнтованого дизайну, систему буде легко супроводжувати та вдосконалювати.

Під час аналізу предметної сфери було висвітлено основні існуючі проблеми ринку програмних та апаратних засобів окулографії, виконано огляд бібліотек комп'ютерного зору та їх порівняння.

Проаналізовано бібліотеки комп'ютерного зору, наведено метод відстеження відповідності фокусу зору зі структурою інформації на моніторі, а також розроблено комплексну модель користувацького інтерфейсу додатка.

Інтерфейс програми було спроектовано з урахуванням потреб цільового користувача. Було досягнуто основної мети – зробити інтерфейс зручним та зрозумілим.

Було проведено декілька експериментів за різних умов використання однієї або двох камер, використання різних налаштувань процесу калібрування, зміна положення додаткової відеокамери, стан голови користувача під час експерименту.

На підставі проведених експериментів дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі було встановлено, що похибка розпізнавання фокусу зору становить 14% по обом осям.

Було виявлено можливі небезпечні та шкідливі виробничі фактори на робочому місці програміста-розробника та надані рекомендації щодо їх усунення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1 Снігур Ю.О., Разносілін В.В. Отслеживание фокуса зрения пользователя при работе с компьютером: тез. XIII Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (11.12-12.12.2019) / Дніпровський національний університет залізничного транспорту академіка В. Лазаряна, 2019.
- 2 Снігур Ю.О., Розробка засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК: тез. . XIV Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (15.12-16.12.2020) / Дніпровський національний університет залізничного транспорту академіка В. Лазаряна, 2020.
- 3 Стаття з Web-сайту Строеие и функции глаза [Електронний ресурс]. – Режим доступу: <https://excimerclinic.ru/press/stroenieglaza/> (дата звернення: 05.09.2020).
- 4 Аналіз сучасного стану дослідження окулографії [Електронний ресурс]. – Режим доступу: <https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BA%D0%BA%D0%B0%D0%B4%D0%B0> (дата звернення: 05.09.2020).
- 5 Матеріал з Вікіпедії — доступна енциклопедія. Окулографія [Електронний ресурс]. – Режим доступу :<https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%83%D0%BB%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F> (дата звернення: 06.09.2020).
- 6 История айтрекинга в лицах и картинках. [Електронний ресурс]. – Режим доступу: <https://usabilityin.ru/eye-tracking-history/> (дата звернення: 06.09.2020).
- 7 Стаття з Web-сайту. Yabus, eye movements, and vision. В W Tatler, N J Wade, H Kwan, J M Findlay, В M Velichkovsky Published under a Creative Commons Licence a Pion publication. Copyright 2010 – Режим доступу: <https://journals.sagepub.com/doi/pdf/10.1068/i0382> (дата звернення: 05.09.2020).

- 8 Барабанщиков, В. А. Методы регистрации движений глаз: теория и практика [Текст] / В. А. Барабанщиков, А. В. Жегалло // Электронный журнал «Психологическая наука и образование». – 2010. – № 5. – С. 240–252. (дата звернения: 07.09.2020).
- 9 Костин, А. Современные ай-трекеры и их возможности для юзабилити-тестирования. Часть II [Электронный ресурс]. – Режим доступа: <https://usabilitylab.ru/blog/obzor-modelej-aj-trekerov/> (дата звернения: 07.09.2020).
- 10 Айтрекинг. Методическое пособие по применению [Текст] / [сост. Е.Ю.Шелепин, К.Ю.Шелепин, К.А.Скуратова]. – СКИФИЯ-ПРИНТ, 2019. – 54 с. (дата звернения: 07.09.2020).
- 11 Understanding different aspects of learning [Электронный ресурс]. – Режим доступа: <https://www.tobiipro.com/applications/scientific-research/education/> (дата звернения: 07.09.2020).
- 12 Halszka Jarodzka, Кеннет Холмквист. Айтрекинг в образовательной науке: теоретические основы и исследовательские программы. [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/316633908_Eye_tracking_in_Educational_Science_Theoretical_frameworks_and_research_agendas (дата звернения: 16.09.2020).
- 13 Милках Луханю. Устройства слежения за глазами могут улучшить системы образования. [Электронный ресурс]. – Режим доступа: <https://techmoran.com/2019/11/12/eye-tracking-devices-can-improve-education-systems/> (дата звернения: 17.09.2020).
- 14 David Rosengrant, Doug Hearrington, Kerriann Alvarado and Danielle Keeble. Following Student Gaze Patterns in Physical Science Lectures. [Электронный ресурс]. – Режим доступа: https://www.tobiipro.com/siteassets/tobii-pro/customer-cases/tobii_customercasepaper_followingstudentgazepatterns_in_physicalsciencelectures.pdf?v=1.0 (дата звернения: 17.09.2020).
- 15 Jonathan Oglesby, project leader, Department of Geography and Geology. Eye Tracking and Water Education in West Africa. [Электронный ресурс]. – Режим доступа:

- <https://www.tobiipro.com/applications/scientific-research/education/customer-cases/western-kentucky-university/> (дата звернення: 17.09.2020).
- 16 Булатников, Е.В. Сравнение библиотек компьютерного зрения для применения в приложении, использующем технологию распознавания плоских изображений [Текст] / Е.В. Булатников, А.А. Гоева // Вестник МГУП имени Ивана Федорова. – М. МГУП им. Ивана Федорова, 2015 г. – С. 85-91. (дата звернення: 27.09.2020).
- 17 Шахин Гадир Сравнительный анализ библиотек компьютерного зрения. – Access Mode : DOI: 10.24411/2520-6990-2019-10812/ «Colloquium-journal» № 24(48), 2019 / ARCHITECTURE. (дата звернення: 28.09.2020).
- 18 Стаття з Web-сайту Аннотации лицевых точек [Электронный ресурс]. – Режим доступа: <https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/> (дата звернення: 28.09.2020).
- 19 Стаття з Web-сайту Tracking your eyes with Python [Электронный ресурс]. – Режим доступа: <https://medium.com/@stepanfilonov/tracking-your-eyes-with-python-3952e66194a6> (дата звернення: 28.09.2020).
- 20 Стаття з Web-сайту Обработка растровых изображений [Электронный ресурс]. – Режим доступа: <http://pzs.dstu.dp.ua/ComputerGraphics/raster/index.html> (дата звернення: 28.09.2020).
- 21 SATOSHI SUZUKI, Topological Structural Analysis of Digitized Binary Images by Border Following. / SATOSHI SUZUKI, KEIICHI ABE // COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING N,32-46 (1985) – Access Mode : <http://pdf-s3.xuebalib.com:1262/xuebalib.com.17233.pdf> Date of Access: 28.09.2020.
- 22 Emo Velzl, Smallest enclosing disks (balls and ellipsoids) [Virtual Resource] / Emo Velzl // Appeared in «New Results and New Trends in Computer Science», (H. Maurer, Ed.) Lecture Notes in 555 (1991) 359-370. – Access Mode : https://people.inf.ethz.ch/emo/PublFiles/SmallEnclDisk_LNCS555_91.pdf. Date of Access: 29.09.2020.

23 Айтрекинг в психологической науке и практике [Текст] / под ред. В.А. Барабанщикова. – Москва : Когито-Центр, – 2016. – 408 с.

24 Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University in Prague: Real-Time Eye Blink Detection using Facial Landmarks / Tereza Soukupova and Jan Čech – Center for Machine Perception: 21st Computer Vision Winter Workshop Luka Cehovin, Rok Mandeljc, Vitomir Struc (eds.) . – Rimske Toplice, Slovenia, February 3–5, 2016. – Access Mode : URL : <http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf> – Title from Screen. – Date of Access: 29 09. 2020.

25 Матеріал з Вікіпедії — доступна енциклопедія. Метод найменших квадратів [Електронний ресурс]. – Режим доступу

https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BD%D0%B0%D0%B9%D0%BC%D0%B5%D0%BD%D1%88%D0%B8%D1%85_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D1%96%D0%B2 (дата звернення: 30.09.2020).

26 Стаття з Web-сайту Розробка uml діаграми варіантів використання [Електронний ресурс].– Режим доступу:

<https://studfile.net/preview/5200239/page:6/#:~:text=%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0%20%D0%B2%D0%B0%D1%80%D1%96%D0%B0%D0%BD%D1%82%D1%96%D0%B2%20%D0%B2%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D0%BD%D1%8F%20%E2%80%93%20%D1%86%D0%B5%20%D0%B3%D1%80%D0%B0%D1%84,%D1%96%20%D0%B2%D1%96%D0%B4%D0%BD%D0%BE%D1%81%D0%B8%D0%BD%20%D0%BC%D1%96%D0%B6%20%D1%86%D0%B8%D0%BC%D0%B8%20%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BC%D0%B8>. (дата звернення: 30.09.2020).

27 Стаття з Web-сайту Архитектура MVC [Електронний ресурс].– Режим доступу: <https://github.com/codedokode/pasta/blob/master/arch/mvc.md> дата звернення: 30.09.2020).

- 28 Стаття з Web-сайту Об'єктно-орієнтоване програмування [Електронний ресурс].– Режим доступу: <http://ruslan.rv.ua/python-essential/methodologies/oop/> дата звернення: 30.09.2020).
- 29 Стаття з Web-сайту Діаграми класів [Електронний ресурс].– Режим доступу: <https://it.wikireading.ru/27850> дата звернення: 30.09.2020).
- 30 ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», затвердженими Постановою головного державного санітарного лікаря України 10 грудня 1998 р. № 7.
- 31 ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки, затверджено та введено в дію наказом міністерства економічного розвитку і торгівлі України 14.10.2013 № 1231.
- 32 ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення.
- 33 ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища, введено в дію з 01.01.2006.
- 34 НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», затвердженими Наказом Міністерства соціальної політики України 14.02.2018 № 207.
- 35 ДБН В.2.5-28:2018 Державні будівельні норми України «Природне і штучне освітлення», затверджені наказом Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 03.10.2018 № 264, введено в дію з 01.03.2019.
- 36 ДСН 3.3.6.042-99 «Державні санітарні норми мікроклімату виробничих приміщень», затверджені Постановою головного санітарного лікаря України № 42 від 1 грудня 1999 року.

37 ДСН 3.3.6.037-99 «Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку», затверджені Постановою головного санітарного лікаря України № 37 від 1 грудня 1999 року.

38 Закон України «Про охорону праці» зі змінами і доповненнями, введений в дію Постановою ВР № 2695-XII від 14.10.92, ВВР, 1992, № 49, ст.669.

39 «Керівництво з визначення і оцінки важкості та напруженості трудового процесу» НАМН України від 02.05.2019р.

40 ДСанНіП «Гігієнічна класифікація праці», затверджені Наказом Міністерства охорони здоров'я України №248 від 08.04.14р.

41 НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні», затверджені Наказом Міністерства внутрішніх справ України 30 грудня 2014 року N 1417, зареєстровані в Міністерстві юстиції України 05 березня 2015 р. за N 252/26697.

42 Кодекс цивільного захисту України зі змінами та доповненнями, Відомості Верховної Ради (ВВР), 2013, № 34-35, ст.458.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Дніпровського
національного університету
залізничного транспорту імені
академіка В. Лазаряна

_____ Б. Є. Боднар

АВТОМАТИЗОВАНА СИСТЕМА ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ
ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ КОМП'ЮТЕРА

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

01116130.01192-01-ЛЗ

Завідувач кафедри КІТ

_____ В. І. Шинкаренко

Керівник розробки

_____ В. І. Шинкаренко

Виконавець

_____ Ю.О. Снігур

Нормоконтролер

_____ О. С. Куроп'ятник

ЗАТВЕРДЖЕНО

01116130.01192-01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ
ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ КОМП'ЮТЕРА

Технічне завдання

01116130.01192-01

Листів 27

2020

АНОТАЦІЯ

Документ 01116130.01192-01 «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера. Технічне завдання» входить до складу програмної документації до програми, яка розпізнає фокус зору користувача, шляхом обробки кадрів, отриманих з відеокамери та шукає відповідність отриманих координат фокусу зору зі структурою інформації на моніторі.

У даному документі представлене призначення та область застосування програмного продукту, основні вимоги, стадії та строки виконання проекту, технічні та техніко-економічні показники, що пред'являються до програмного продукту.

ЗМІСТ

Вступ.....	4
1 Підстава для розробки	5
2 Призначення розробки.....	6
2.1 Функціональне призначення	6
2.2 Експлуатаційне призначення	6
3 Вимоги до програми.....	7
3.1 Вимоги до функціональних характеристик.....	7
3.2 Вимоги до надійності.....	8
3.3 Умови експлуатації	9
3.4 Вимоги до складу і параметрів технічних засобів.....	9
3.5 Вимоги до інформаційної та програмної сумісності.....	9
3.6 Вимоги до маркування і упаковки.....	9
3.7 Вимоги до транспортування і зберігання	10
4 Вимоги до програмної документації	11
5 Визначення витрат на проектування програми.....	12
6 Стадії та етапи розробки.....	22
7 Порядок контролю та прийому	23
Бібліографічний список	24

ВСТУП

На сьогоднішній день, у зв'язку з поширенням епідемії коронавірусу Covid-19, закриваються навчальні заклади по всьому світу. В цей час надзвичайно зростає потреба у дистанційному навчанні. Студентам важко концентрувати увагу, коли вони засвоюють матеріал самостійно. Тому актуальним є завдання покращення концентрації уваги студента під час дистанційного навчання. Для вирішення цього завдання дослідники проводять дослідження, щоб відстежити на що студент звертає увагу, що залишається поза увагою, скільки часу він витрачає на засвоєння того чи іншого матеріалу. Для проведення таких досліджень необхідні спеціальні засоби відстеження відповідності фокусу зору зі структурою інформації на моніторі.

Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера дозволяє зробити висновки щодо необхідності редагування інформаційного контенту, для покращення концентрації уваги студента під час дистанційного навчання.

Основна термінологія: КАЛІБРУВАННЯ, ФОКУС ЗОРУ, РОЗПІЗНАВАННЯ ОБРАЗІВ.

Причинами виникнення є відсутність аналогів у відкритому просторі.

Область застосування: навчальні заклади.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ ректора Дніпровського національного університету залізничного транспорту імені академіка В. Лазаряна проф. Пшінька О. М. №779ст від 10.10.2019р. «Про призначення наукових керівників та затвердження тем магістерських робіт» факультету «Технічна кібернетика» за спеціальністю 121 «Інженерія програмного забезпечення».

Тема роботи: розробка методів дослідження відстеження відповідності фокусу зору зі структурою інформації на моніторі, керівник розробки проф. Шинкаренко В.І.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Функціональним призначенням розробки є відстеження фокусу зору користувача та пошук відповідності зі структурою інформації на моніторі. Програма не призначена для відстеження фокусу зору кількох користувачів одночасно.

2.2 Експлуатаційне призначення

Експлуатаційне призначення полягає у відстеженні відповідності фокусу зору, задля дослідження поведінки користувача під час роботи з інформаційним контентом. Результат дослідження дозволить зрозуміти, на які елементи інформаційного контенту зосереджено більше уваги, а які залишилися поза увагою, скільки часу знадобилося для засвоєння даного матеріалу, тощо. Дану інформацію можна використовувати для налаштування навчального контенту та можливості адаптувати, налаштовувати та оптимізувати процес навчання.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик програмного забезпечення:

- відстеження координат фокусу зору користувача в області екрану монітора під час роботи за комп'ютером. Відстеження відбувається за рахунок знаходження зміщення центру зіниці ока відносно центру ока;
- використання декількох відеокамер для розпізнавання руху зіниць під різним кутом;
- забезпечення процесу калібрування задля підвищення точності розпізнавання фокусу зору;
- встановлення інформаційного контенту на моніторі та відстеження відповідності отриманих координат фокусу зору зі структурою інформації на екрані. Знайдена відповідність зберігається у файл для подальшого аналізу поведінки користувача під час роботи з інформаційним контентом.

Вхідними даними програми є:

- параметри налаштування алгоритму розпізнавання зіниці ока: розмір апертури медіанного фільтр, порогове значення;
- параметри налаштування процесу калібрування: форма калібрувальної точки, колір, таймер затримки позиції калібрувальної точки у секундах, розмір калібрувальної точки, кількість позицій калібрувальної точки по осі ординат та абсцис;
- параметри налаштування інформаційного контенту на моніторі: шлях до директорії;
- потік відеокадрів зі зображенням користувача, відзнятих відеокамерою під час роботи користувача з комп'ютером.

Результатом роботи програми є наступні вихідні дані:

- дані, зібрані під час процесу калібровки. Файл повинен містити наступні дані: координати позиції калібрувальної точки, координати контуру лівого та правого ока, координати центру зіниці лівого та правого ока;
- дані зібрані під час співвідношення координат фокусу зору з інформаційним контентом на моніторі: точка фокусу зору, ім'я елементу інформаційного контенту, область розташування елементу інформаційного контенту на моніторі;
- зображення інформаційного контенту на моніторі з відображенням координати фокусу зору у вигляді червоної точки.

3.2 Вимоги до надійності

Надійність системи повинна бути забезпечена за наступними напрямками:

- забезпечення працездатності компонентів програмно-технічної платформи;
- збереження даних.

При цьому повинна вимагатися мінімальна увага з боку системного адміністратора щодо реакції на усунення наслідків відмов компонентів, а також програмно-апаратними засобами повинно бути забезпечене збереження даних. ПЗ повинне забезпечувати відмовостійку роботу в режимі 24x7x365 і гарантувати доступність для роботи кінцевих користувачів на рівні мінімум 99%.

Максимальний час відновлення працездатності ПЗ не більше 30 хвилин. Збереженість інформації на випадок аварій повинна бути забезпечена у повному обсязі. Резервне копіювання має відбуватися з періодичністю, що забезпечує повне збереження та відновлення даних.

Збереження даних має забезпечуватися у випадках:

- вимкнення живлення;
- відмови технічних засобів обробки інформації;
- помилки, збоїв або руйнування програмного забезпечення. Вимоги щодо надійності

Системи можуть бути уточнені Виконавцем та повинні бути зазначені в наступних версіях технічного завдання.

3.3 Умови експлуатації

Даний програмний продукт може використовуватись в умовах, які відповідають вимогам документу [1].

Для нормального функціонування програмного продукту необхідно виконання наступних вимог:

- ЕОМ повинна відповідати вимогам чинних в Україні стандартів, нормативних актів з охорони праці [2];
- програма повинна експлуатуватись у приміщенні, призначеному для роботи з ЕОМ, з відповідними кліматичними умовами: температура 21° – 25°C та вологість 40 - 60%;
- стан технічних засобів повинен задовольняти відповідним нормам та вимогам;

Для забезпечення надійного функціонування програмного продукту необхідно дотримуватися таких умов:

- програма призначена для роботи на ЕВМ з операційною системою Windows 10;
- працювати з програмою може людина, яка володіє навичками роботи з ОС у графічному режимі та ознайомила з керівництвом користувача.

3.4 Вимоги до складу і параметрів технічних засобів

Пристрій, що має щонайменше 128Мб вільного місця на вбудованому носії інформації та 1024Мб оперативної пам'яті для коректної роботи. Вбудована відеокамера з роздільною здатністю 1280x720 та додаткова зовнішня відеокамера з роздільною здатністю 1280x1024. Для зручної роботи з програмою необхідно мати монітор з роздільною здатністю 1366x768, маніпулятор «миша» та клавіатуру. Для встановлення ПЗ необхідна наявність USB роз'єму, або підключення до мережі Інтернет.

3.5 Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційної та програмної сумісності: ОС Windows.

3.6 Вимоги до маркування і упаковки

Програмний продукт може маркуватися як зображено на рис. 3.1:

Автоматизована система відстеження відповідності
фокусу зору зі структурою інформації на моніторі
комп'ютера, 1.0
Снігур Юлія Олександрівна
кафедра КІТ, ДНУЗТ 2020

Рисунок 3.1 – Маркувальний штамп

3.7 Вимоги до транспортування і зберігання

Транспортувати програмний продукт можна наступними способами:

- через всесвітню систему взаємополучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів – Інтернет;
- за допомогою портативних носіїв інформації –USB-накопичувачів.

Термін збереження обумовлений збереженням інформації на носії.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації має входити технічне завдання та робочий проект.

До складу робочого проекту мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача. Керівництво відстеження фокусу зору користувача.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів [3].

5 ВИЗНАЧЕННЯ ВИТРАТ НА ПРОЕКТУВАННЯ ПРОГРАМИ

Основна мета розробки техніко-економічного обґрунтування (ТЕО) – дати фінансову оцінку передбачуваних витрат та одержуваного корисного результату, а також оцінити прибутковість проекту і, в кінцевому підсумку, економічну доцільність його розробки та впровадження.

Початковим етапом розрахунку величини трудових витрат розробників є оцінка розміру програмного забезпечення. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості (кількісний або якісний) [4].

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \times S^b \times EAF, \quad (5.1)$$

де E – витрати праці на проект (в людино-місяцях);

S^b – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, $a = 2,4$; $b = 1,05$.

Розмір програмного коду було підраховано за допомогою інструменту командного рядка для сканування папок для вихідних файлів коду і підрахунку кількості рядків вихідних кодів в ньому – Pygount (рис. 5.1).

Отже, розмір програмного коду становить 2301 рядок:

$$E = 2,4 \times 2,301^{1,05} \times 1 = 5,76.$$


```

A:\University\eye_tracking>pygount --format=summary src
  Language      Files      %      Code      %      Comment      %
  -----
Python          51     63.75    2266     98.48      141     99.30
RPMSpec           1      1.25      32      1.39         1      0.70
JSON              3      3.75       3      0.13         0      0.00
__unknown__       3      3.75       0      0.00         0      0.00
__empty__        16     20.00       0      0.00         0      0.00
__duplicate__     1      1.25       0      0.00         0      0.00
__binary__        5      6.25       0      0.00         0      0.00
  -----
Sum total        80                      2301                      142

```

Рисунок 5.1 – Підрахунок розміру програмного коду

Отже, згідно моделі COSOMO, орієнтовні трудовитрати на проект складуть приблизно 5,76 людино-місяці.

Нижче наведені розрахунки вартості розробки «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі». Основними статтями витрат прийняті:

- основна заробітна плата;
- відрахування на соціальні потреби;
- накладні витрати;
- витрати на персональний комп'ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину. Розрахунок заробітної платні проводиться по формі табл. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол.	місяців	
1	інженер-програміст	10 837 [5]	1	5,76	77184,00

Описаний в проекті програмний продукт був розроблений одним програмістом в період з 13.01.20 до 19.06.20, що складає 115 днів з розрахунку (15 – у січні, 20 – у лютому, 21 – у березні, 21 – у квітні, 19 – у травні та 19 – у червні) та 6 вихідних днів (1 – у березні, 1 – у квітні, 2 – у травні, 1 – у червні) або 23 робочих тижні. Витрати робочого часу прийняті за 40 годин у тиждень (168 годин у місяць), оплата за святкові дні нараховується як за робочий день. Погодинна ставка кваліфікованого інженера–програміста складає 64 грн/год ($10\,837/168=64$ грн). Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} \times N_{\text{тиж}} \times N_{\text{год}}, \quad (5.2)$$

де $N_{\text{чол}}$ – кількість виконавців, *чол.*;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, *год.*

$$t_{\text{розробки}} = 1 \times 23 \times 40 = 920 \text{ чол/год.}$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \times N \times K_{KB}, \quad (5.3)$$

де $t_{\text{розробки}}$ – витрати праці у чол/год;

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, приймається 0,75.

ОЗП складає:

$$\text{ОЗП} = 920 \times 64 \times 0,75 = 44\,160 \text{ грн.}$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати та на сьогоднішній день складає 22% [6]:

$$C_{\text{соц}} = \frac{\text{ОЗП} \times 22\%}{100\%}. \quad (5.4)$$

$$C_{\text{соц}} = \frac{44160 \times 22\%}{100\%} = 9715 \text{ грн.}$$

Отримані результати за (5.3) та (5.4) підсумовуються. Вони складають 53 875 грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30 – 40%) від суми прямих витрат:

$$C_{\text{накл}} = \frac{(\text{ОЗП} + C_{\text{соц}}) \times 35\%}{100 \%}. \quad (5.5)$$

$$C_{\text{накл}} = \frac{(44\,160 + 9715) \times 35\%}{100 \%} = 18\,856 \text{ грн.}$$

На протязі усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов'язані з її експлуатацією.

Експлуатаційні витрати на персональний комп'ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп'ютеру. В експлуатаційні витрати входять:

- вартість витратних матеріалів;

- витрати на ремонт;
- заробітна плата ремонтника;
- оренда приміщення;
- додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;
- амортизаційні витрати на персональний комп'ютер і програмне забезпечення.

Витрати на електроенергію ($C_{ел}$) визначаються за формулою:

$$C_{ел} = P \times B \times T_{розр}, \quad (5.6)$$

де P – потужність комп'ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год [7];

B – вартість 1 кВт/година, складає 2,73 грн [8];

$T_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{ел} = 0,45 \times 2,73 \times 920 = 1130 \text{ грн.}$$

Витрати на витратні матеріали ($C_{вм}$) протягом всього терміну експлуатації приблизно 10 % від вартості комп'ютеру. Вартість робочої станції приймається 26 555 грн. [9], термін експлуатації – 5 років. Отже, можна визначити ці витрати за період створення програмного засобу:

$$C_{вм} = B_{ком} \times \frac{N_{д}}{N_{експ} \times 365} \times \frac{10 \%}{100 \%}, \quad (5.7)$$

де $B_{ком}$ – вартість персонального комп'ютеру;

N_D – кількість днів розробки програмного продукту;

$N_{\text{експ}}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{\text{вм}} = 26555 \times \frac{115}{5 \times 365} \times \frac{10 \%}{100 \%} = 167 \text{ грн.}$$

Заробітна плата ремонтника ($C_{\text{рем}}$) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 10 000 грн [10]. Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{\text{рем}} = \frac{C_{\text{рем}}}{N_{\text{КОМ}}} \times T_{\text{міс}}, \quad (5.8)$$

де $C'_{\text{рем}}$ – середньомісячна заробітна плата;

$N_{\text{КОМ}}$ – кількість комп'ютерів на одного ремонтника.

$T_{\text{міс}}$ – час розробки програмного продукту, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{10000}{50} \times 5,76 = 1152 \text{ грн.}$$

За статистикою витрати на комплектуючі вироби ($C_{\text{КОМ}}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{КОМ} = C_{ВМ} = 167 \text{ грн.} \quad (5.9)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АКП} = B_{КОМ} \times \frac{N_{Д}}{N_{\text{експ}} \times 365}. \quad (5.10)$$

$$\text{АКП} = 26555 \times \frac{5,76}{3 \times 12} = 4248.$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийняти термін морального старіння для Windows 5 років та PyCharm за 1 рік то амортизаційні відрахування на програмне забезпечення дорівнюють його вартості.

Для функціонування персонального комп'ютера використовувалася операційна система Windows 10, для написання програмного забезпечення – програмне середовище PyCharm.

$$\text{АПЗ}_{\text{Windows}} = 6918 \times \frac{5,76}{5 \times 12} = 664,13.$$

$$\text{АПЗ}_{\text{PyCharm}} = 5611 \times \frac{5,76}{1 \times 12} = 2693,28.$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10 Professional	6918	http://mtsoft.kiev.ua/product/windows-10-professional	664
PyCharm	5611	https://www.jetbrains.com/ru-ru/pycharm/buy/#commercial?billing=yearly	2693
Всього:	12529	-	3357

Додаткові витрати ($C_{\text{дод}}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера-системотехніка, тобто 5000 гривень на місяць.

Оренду приміщень для однієї людини приймемо рівною 2 766,67 гривень на місяць (загальна вартість оренди приміщення (21 м^3) на 3 інженерів-програмістів 8 500 грн. на місяць [11]). Тобто за весь період розробки – 15 936,00 грн.

Сумарні експлуатаційні витрати на один персональний комп'ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АПК} + \text{АПЗ} + C_{\text{ор}} + C_{\text{дод}}. \quad (5.11)$$

$$C_{\text{експ}} = 1130 + 167 + 1152 + 4248 + 3357 + 15936 + 5000 = 30\,990 \text{ грн.}$$

Результати розрахунків зведено у табл. 5.3.

Найменування витрат	Витрати, грн
Витрати на електроенергію	1 130
Вартість витратних матеріалів	167
Витрати на ремонт	1 152
Комплектуючі вироби	167
Амортизація персонального комп'ютера	4 248
Амортизація програмного забезпечення	3357
Оренда приміщення	15 936
Додаткові витрати	5 000
Всього	30 990

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \quad (5.12)$$

$$C_{\text{розробки}} = 44\,160 + 9\,715 + 18\,856 + 30\,990 = 103\,721 \text{ грн.}$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	44 160
Відрахування на соціальні потреби	9 715
Накладні витрати	18 856
Експлуатаційні витрати	30 990
Всього	103 721

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера. За результатами розрахунків, приблизна вартість розробки складає 103 721 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки приводяться в табл. 6.1.

Таблиця 6.1 – Стадії та етапи розробки

№	Стадії розробки	Етап розробки	Термін
1	Технічне завдання	Постановка задачі	01.10.19 – 30.12.19
		Розробка структур вхідних і вихідних даних	14.01.20 – 30.01.20
		Розробка вимог до програми	03.02.20 – 28.02.20
		Затвердження технічного завдання	02.03.20 - 31.03.20
2	Робочий проект	Розробка і програмування логіки програми	01.04.20 – 30.06.20
		Розробка і програмування користувацького інтерфейсу	01.07.20 – 7.08.20
		Відлагодження програми	10.08.20 – 31.08.20
		Розробка програмної документації	01.09.20 – 30.10.20
3	Впровадження	Підготовка і передача програми і програмної документації для супроводу	01.11.20 – 12.12.20

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙОМУ

Контроль здійснюється за допомогою виконання набору тестів з метою знаходження помилок в програмному продукті та його специфікації. Контроль виконання роботи забезпечується головним керівником розробки.

Прийом програмного продукту здійснюється уповноваженою комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».
2. Закон Міністерства охорони здоров'я України від 09.10.2000 № 247 (у редакції наказу МОЗ від 14.03.2006 № 120) «Про затвердження Тимчасового порядку проведення державної санітарно-гігієнічної експертизи».
3. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. – 38 с.
4. Методики оценки трудозатрат по разработке программного обеспечения информационных систем [Текст]: Є. Борисенков; ТарГУ ім. М.Х.Дулати.
5. Огляд статистики зарплатні професії "Інженер-програміст в Україні" [Електронний ресурс]. – Режим доступу: <https://ua.trud.com/ua/salary/2/67643.html> (дата звернення: 10.10.2020).
6. Закон України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування», стаття 8, пункт 5. Режим доступу: <https://zakon.rada.gov.ua/laws/show/2464-17#Text>.
7. Стаття з Web-сайту Середня потужність комп'ютера [Електронний ресурс]. – Режим доступу: <https://beasthackerz.ru/uk/windows-7/srednyaya-moshchnost-kompyutera-vt-skolko-elektroenergii-potreblyaet.html> (дата звернення: 10.10.2020).
8. Офіційний сайт Національної комісії, що здійснює державне регулювання у сферах енергетики та комунальних послуг. Тарифи на електроенергію для непобутових споживачів. Режим доступу: <http://www.nerc.gov.ua/>.
9. Ноутбук Asus ROG Strix G15 G512LI-HN094. Режим доступу: https://rozetka.com.ua/asus_90nr0381_m01620/p231822349/.

10. Системний адміністратор, сервісний інженер. Пошук робочих місць. Режим доступу:
<https://www.work.ua/ru/resumes/425938/>

11. Об'ява «Аренда офисов -21, 9, 11, 31, 19 м.кв. центр города, собственник». Режим доступу:
<https://www.olx.ua/uk/obyavlenie/arenda-ofisov-21-9-11-31-19-m-kv-tsentr-goroda-sobstvennik-IDm1cOs.html#fd7f062b72;promoted>

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Дніпровського
національного університету
залізничного транспорту імені
академіка В. Лазаряна

_____ Б. Є. Боднар

АВТОМАТИЗОВАНА СИСТЕМА ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ
ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ КОМП'ЮТЕРА

Робочий проєкт

ЛИСТ ЗАТВЕРДЖЕННЯ

01116130.01192-01-ЛЗ

Завідувач кафедри КІТ

_____ В. І. Шинкаренко

Керівник розробки

_____ В. І. Шинкаренко

Виконавець

_____ Ю.О. Снігур

Нормоконтролер

_____ О. С. Куроп'ятник

ЗАТВЕРДЖЕНО

01116130.01192-01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ
ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ КОМП'ЮТЕРА

Специфікація

01116130.01192-01

Листів 2

Позначення	Найменування	Примітка
	<u>Документація</u>	
01116130.01192-01-ЛЗ	Лист затвердження	
01116130.01192-01 12 01-ЛЗ	Лист затвердження	
01116130.01192-01 12 01	Текст програми	
01116130.01192-01 13 01-ЛЗ	Лист затвердження	
01116130.01192-01 13 01	Опис програми	
01116130.01192-01 ІЗ 01-ЛЗ	Лист затвердження	
01116130.01192-01 ІЗ 01	Керівництво користувача. Керівництво відстеження фокусу зору користувача	

ЗАТВЕРДЖЕНО

01116130.01192-01 13 01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ
ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ КОМП'ЮТЕРА

Опис програми

01116130.01192-01 13 01

Листів 24

АНОТАЦІЯ

Документ 01116130.01192-01 13 01 «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера. Опис програми» входить до складу програмної документації до програми.

Програмний продукт дозволяє розпізнавати фокус зору користувача, шляхом обробки кадрів, отриманих з відеокамери та шукає відповідність отриманих координат фокусу зору зі структурою інформації на моніторі. В документі міститься опис програми та її функціональних можливостей. Програма реалізована на мові Python у програмному середовищі PyCharm.

ЗМІСТ

1 Загальні відомості	4
2 Функціональне призначення	5
3 Опис логічної структури.....	6
3.1 Алгоритм програми.....	6
3.2 Використані методи	6
3.3 Структура програми з описом функцій складових частин і зв'язки.....	7
3.4 Зв'язки програми з іншими програмами	11
4 Використані технічні засоби	12
5 Виклик та завантаження	13
6 Вхідні дані.....	14
7 Вихідні дані.....	15
8 Опис призначеного для користувача інтерфейсу	16
8.1 Опис станів програми	16
8.2 Опис керування діалогом	17
9 Порядок роботи з програмою.....	22
10 Повідомлення.....	23

1 ЗАГАЛЬНІ ВІДОМОСТІ

Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера дозволяє зробити висновки щодо необхідності редагування інформаційного контенту для покращення концентрації уваги студента під час дистанційного навчання.

Для експлуатації програмного продукту необхідний будь-який пристрій з параметрами, які задовольняють нормальному функціонуванню операційної системи Windows. Програма реалізована на мові Python у програмному середовищі PyCharm.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціональним призначенням розробки є відстеження фокусу зору користувача та пошук відповідності зі структурою інформації на моніторі. Програма не призначена для відстеження фокусу зору кількох користувачів одночасно.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм програми

Алгоритм автоматизованої системи відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера складається з наступних кроків:

Крок 1: отримання потоку відеокадрів з однієї або двох камер.

Крок 2: обробка відеокадрів. Для кожного кадру розпізнається образ обличчя для пошуку області розташування очей на кадрі.

Крок 3: для кожного ока знаходиться координата його центру та зіниці. Розраховується відстань зміщення зіниці ока відносно його центру.

Крок 4: калібрування. Відбувається збір даних співвідношення координат позиції калібрувальної точки зі обчисленим зміщенням зіниці ока відносно його центру. Дані зберігаються та аналізуються для знаходження залежності між екранними координатами та переміщення зіниці.

Крок 5: налаштування інформаційного контенту на екрані монітора.

Крок 6: підрахунок координат фокусу зору.

Крок 7: пошук відповідності між фокусом зору користувача на інформаційним контентом зображеним на моніторі екрану.

3.2 Використані методи

Програма використовує наступні методи:

- для обробки зображення використовувався медіанний та пороговий фільтр;
- для розпізнавання образу обличчя користувача на відеокадрі використовується алгоритм «Active Shape Models» [1];
- для знаходження залежності координат точки, що спостерігалась користувачем від даних, отриманих під час аналізу відеокадру використовується метод найменших квадратів [2].

3.3 Структура програми з описом функцій складових частин і зв'язки

На рис. 3.1 представлено схему зображення взаємодії модулів системи відстеження відповідності фокусу зору зі структурою інформації на моніторі.

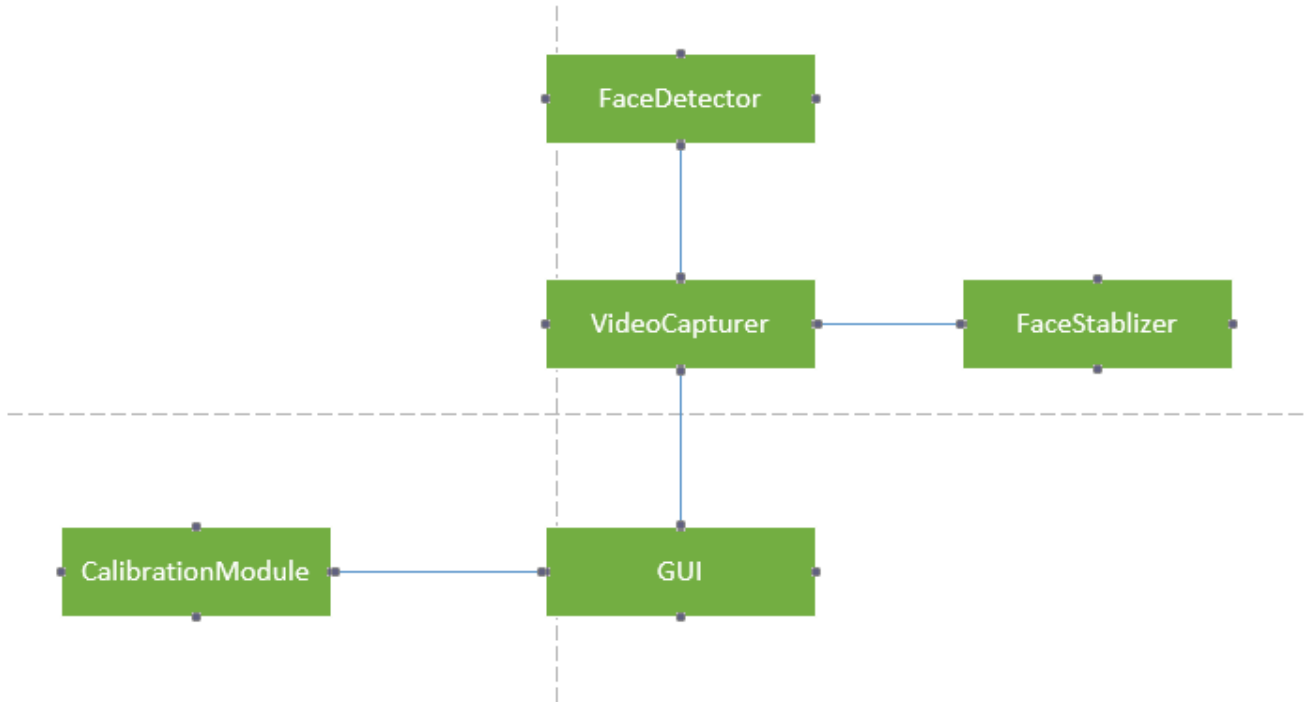


Рисунок 3.1 – Схема взаємодії модулів системи

Пакет «FaceDetector» (рис. 3.2) дозволяє розпізнати образ обличчя на відеокадрі та містить наступні класи:

- FaceDetector – відстеження риси обличчя;
- Face – структура для представлення даних про риси обличчя;
- Eye – структура для представлення даних про око;
- Pupil – структура для представлення даних про зіницю ока;
- EyeDetector – відстеження даних про око на відеофрагменті;
- PupilDetector – відстеження даних про зіницю на відеофрагменті.

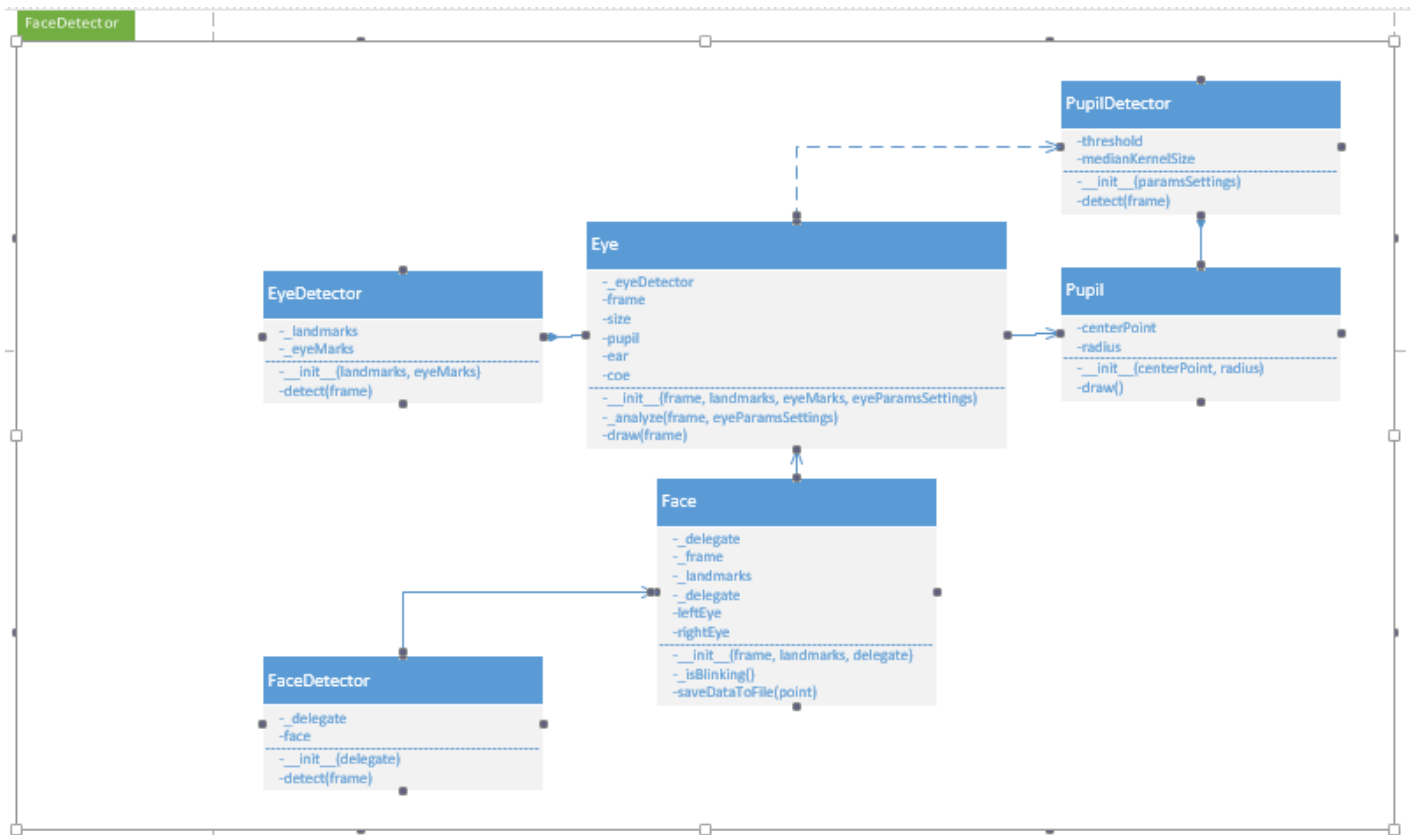


Рисунок 3.2 – Діаграма класів пакета «FaceDetector»

Пакет «FaceStabilizer» (рис. 3.3) дозволяє стабілізувати отримані дані про розпізнані образи обличчя для кількох кадрів (кількість кадрів дорівнює 4), щоб позбутися шумів під час підрахунку координат фокусу зору.

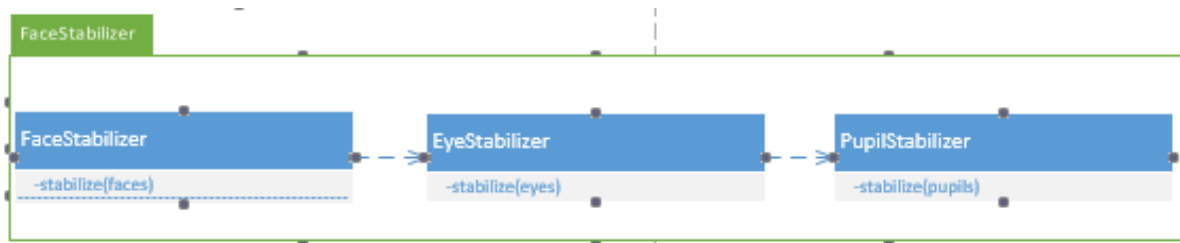


Рисунок 3.3 – Діаграма класів пакета «FaceStabilizer»

Пакет «PointGazeDetector» (рис. 3.4) дозволяє відстежити фокус зору під час роботи з інформаційним контентом.

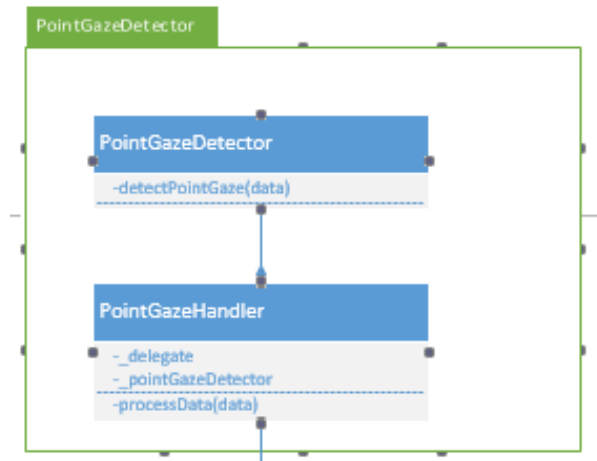


Рисунок 3.4 – Діаграма класів пакета «PointGazeDetector»

Пакет «VideoCapturer» (рис. 3.5) дозволяє отримувати відеокадри з відеокамер.

Клас FrameCapturer дозволяє захватувати кадри з декількох відеокамер одночасно. Клас VideoCapturer дозволяє захватувати кадри з певної відеокамери. Кожний отриманий кадр піддається обробці за допомогою FrameAnalyzer.

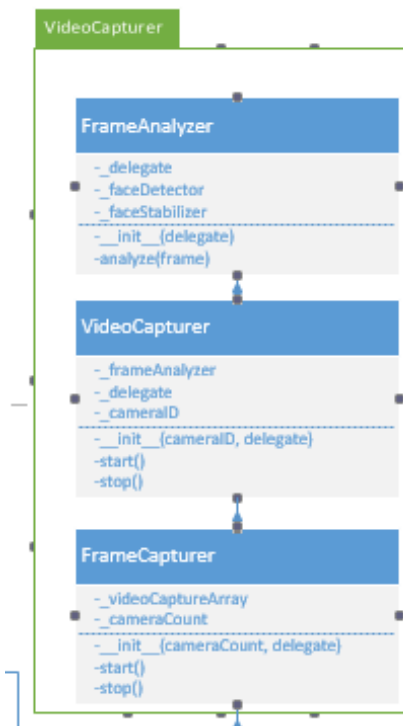


Рисунок 3.5 – Діаграма класів пакета «VideoCapturer»

Пакет «CalibrationModule» (рис. 3.6) відповідає за процес калібрування.

Базовим класом розробки функціоналу процесу калібрування є CalibrationModule. Клас CalibrationSettings представляє собою опції налаштування процесу калібрування. Опції налаштування зчитуються з файлу. Клас CalibrationWidget представляє собою віджет, який з'являється під час виконання процесу калібрування та відображає позиції калібрувальної точки. Після завершення калібрування, отримані дані обробляються за допомогою класу DataAnalyzer.

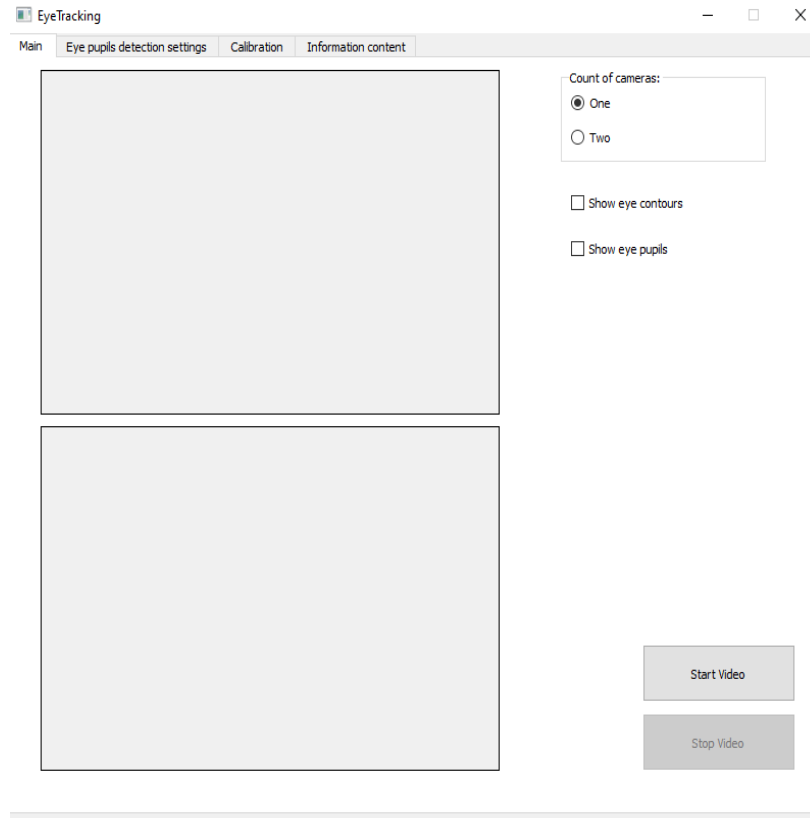


Рисунок 3.6 – Діаграма класів пакета «CalibrationModule»

Пакет «GUI» (рис. 3.7) відповідає за відображення графічного інтерфейсу та обробку сигналів на дії користувача.

Базовим класом ієрархії є MainWindow – клас, який відповідає за обробку сигналів графічного інтерфейсу та є спадкоємцем класу QMainWindow. MainWindow є початковою точкою роботи системи відстеження відповідності фокусу зору зі структурою інформації на моніторі. Для полегшення обробки сигналів графічного інтерфейсу вікон графічного інтерфейсу, було розроблено наступні класи:

LeftEyeSettings, RightEyeSettings, CalibrationTab та InformationContentTab. Кожен клас містить атрибут delegate – посилання на клас MainWindow.

Класи LeftEyeSettings та RightEyeSettings відповідають за обробку сигналів налаштування відстеження лівої та правої зіниці ока. Клас EyeSettings є базовим класом для LeftEyeSettings та RightEyeSettings.

Клас CalibrationTab відповідає за обробку сигналів налаштування процесу калібрування. Клас InformationContentTab відповідає за обробку сигналів налаштування процесу відстеження відповідності фокусу зору зі структурою інформації. За відображення інформаційного конфетну на моніторі відповідає InformationContextWidget.

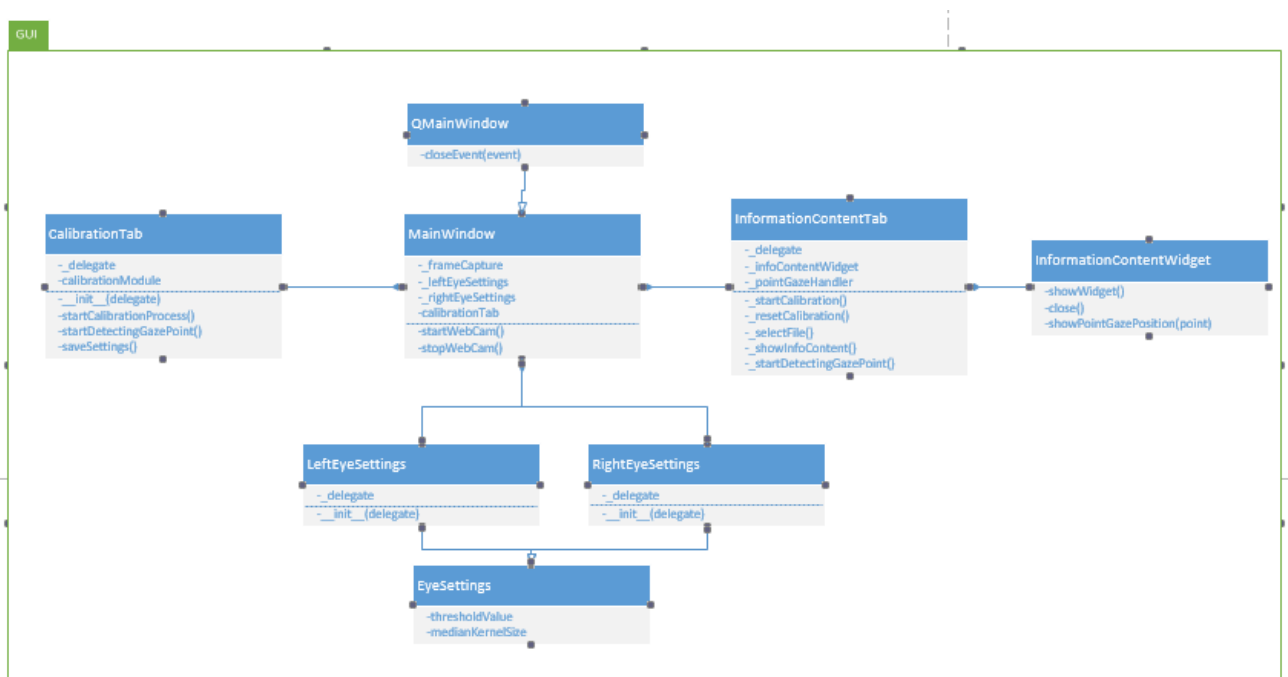


Рисунок 3.7 – Діаграма класів пакета «GUI»

3.4 Зв'язки програми з іншими програмами

Розроблений продукт не має залежностей від сторонніх програми, оскільки в пакеті програми надані необхідні бібліотеки, від яких він залежить.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для експлуатації програмного продукту необхідний будь-який пристрій з параметрами, які задовольняють нормальному функціонуванню операційної системи Windows.

Мінімальна конфігурація пристрою, що забезпечить нормальну роботу програмного продукту:

- щонайменше 128 Мб вільного місця на вбудованому носії інформації;
- щонайменше 1024 Мб оперативної пам'яті;
- монітор з роздільною здатністю 1360x780;
- відеокамера з роздільною здатністю 1280x720;
- додаткова відеокамера з роздільною здатністю 1280x1024;
- центральний процесор з тактовою частотою 1 ГГц та більше;
- маніпулятор «миша»;
- стандартна клавіатура;
- для встановлення ПЗ необхідна наявність USB роз'єму, або підключення до мережі Інтернет.

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Для запуску програмного продукту «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера» необхідно виконати запуск файлу eye_tracking.exe засобами передбаченими встановленою на пристрої ОС.

Об'єм програми складає 20 Мбайт. Комплекс функціонує у середовищі Windows.

6 ВХІДНІ ДАНІ

Вхідними даними програми є:

- кількість камер, що використовуються (1 або 2);
- параметри налаштування алгоритму розпізнавання зіниці ока: розмір апертури для медіанного фільтра (значення у вигляді непарного натурального числа) та значення порогу для порогового фільтра (значення у інтервалі натуральних чисел від 0 до 255);
- параметри налаштування процесу калібрування: форма калібрувальної точки (квадрат або коло), колір (представлений RGB моделлю), таймер затримки позиції калібрувальної точки у секундах (значення у вигляді цілого числа в інтервалі від 0.1 до 100 sec), розмір калібрувальної точки (значення у вигляді натурального числа в інтервалі від 1 до 50 px), кількість позицій калібрувальної точки по осі ординат та абсцис (значення у вигляді натурального числа в інтервалі від 1 до 10);
- параметри налаштування інформаційного контенту на моніторі: шлях до директорії;
- потік відеокадрів зі зображенням користувача, відзнятих відеокамерою під час роботи користувача з комп'ютером.

7 ВИХІДНІ ДАНІ

Результатом роботи програми є наступні вихідні дані:

- дані, зібрані під час процесу калібровки. Файл містить наступні дані: координати позиції калібрувальної точки (значення у вигляді натуральних чисел), координати контуру лівого та правого ока (значення у вигляді натуральних чисел), координати центру зіниці лівого та правого ока (значення у вигляді цілих чисел);
- дані зібрані під час співвідношення координат фокусу зору з інформаційним контентом на моніторі: точка фокусу зору (значення представлене у вигляді двох координат по осі X та Y), ім'я елементу інформаційного контенту, область розташування елементу інформаційного контенту на моніторі (представлена у вигляді координат вершин чотирикутника);
- зображення інформаційного контенту на моніторі з відображенням координати фокусу зору у вигляді червоної точки.

8 ОПИС ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ**8.1 Опис станів програми**

Стани, в яких може знаходитись програма наведено у табл. 8.1.

Таблиця 8.1 – Повідомлення користувачу

№ стану	Стан	Опис стану	Рекомендовані дії
1	Завантаження програми	Програма завантажується до оперативної пам'яті	Почекайте, доки програма запусниться
2	Відкрита програма з незаповненими параметрами	Щойно запущена програма з усіма незаповненими полями	Починайте працювати у програмі
3	Програма у процесі отримання параметрів	Користувач поступово вводить всі параметри	Введіть усі параметри у зручному порядку
4	Програма з заповненими параметрами	Система отримала всі дані, необхідні для роботи	Перейдіть до головного діалогового вікна, та натисніть кнопку «Start»
5	Програма у процесі обробки відеокадрів зі зображенням користувача	Програма обробляє відеокадри та розпізнає образ обличчя, знаходить координати центру правого та лівого ока та їх зіниць. Розраховується відстань зміщення зіниці ока відносно його центру.	Перейдіть до вікна калібрування та натисніть кнопку «Start Calibration». Слідкуйте за переміщенням позиції калібрувальної точки. Намагайтесь не відволікатись, бо це вплине на точність розпізнавання фокусу зору

6	Програма обробляє результати калібровки	Програма зчитує дані з csv файлу та обробляє їх Програма аналізує отримані дані та знаходить залежність між екранними координатами та переміщенням зіниці ока	Почекайте, доки завершиться цей етап. Далі натисніть кнопку «Show a window to full screen»
7	Програма відображає інформаційний контент	Програма розраховує координати фокусу зору користувача та шукає відповідності між інформаційним контентом на екрані. Дані зберігаються у файл	Ознайомтесь з інформаційним контентом

8.2 Опис керування діалогом

Продукт «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера» має декілька діалогових вікон.

Головне вікно з'являється після запуску програми (рис. 8.1). Воно дозволяє обрати кількість відеокамер – одну або дві, які будуть використовуватися для запису користувача та розпочати перехват відеокадрів з камер, натиснувши на кнопку «Start Video». Для візуального відображення контурів очей та зіниць очей на відеокадрах, існують спеціальні опції «Show eye contours», «Show eye pupils». Для зупинки перехвату відеокадрів з камер використовується кнопка «Stop Video».

Вікно «Налаштування відстеження зіниць очей» (рис. 8.2) містить два підвікна, які дозволяють налаштувати процес відстеження зіниці ока для лівого та правого ока. У кожному вікні надається можливість встановити розмір апертури медіанного фільтру та порогове значення порогового фільтру та дослідити поетапний процес відстеження зіниці ока.

Вікно «Калібрування» (рис. 8.3) дозволяє налаштовувати та проводити процес калібрування.

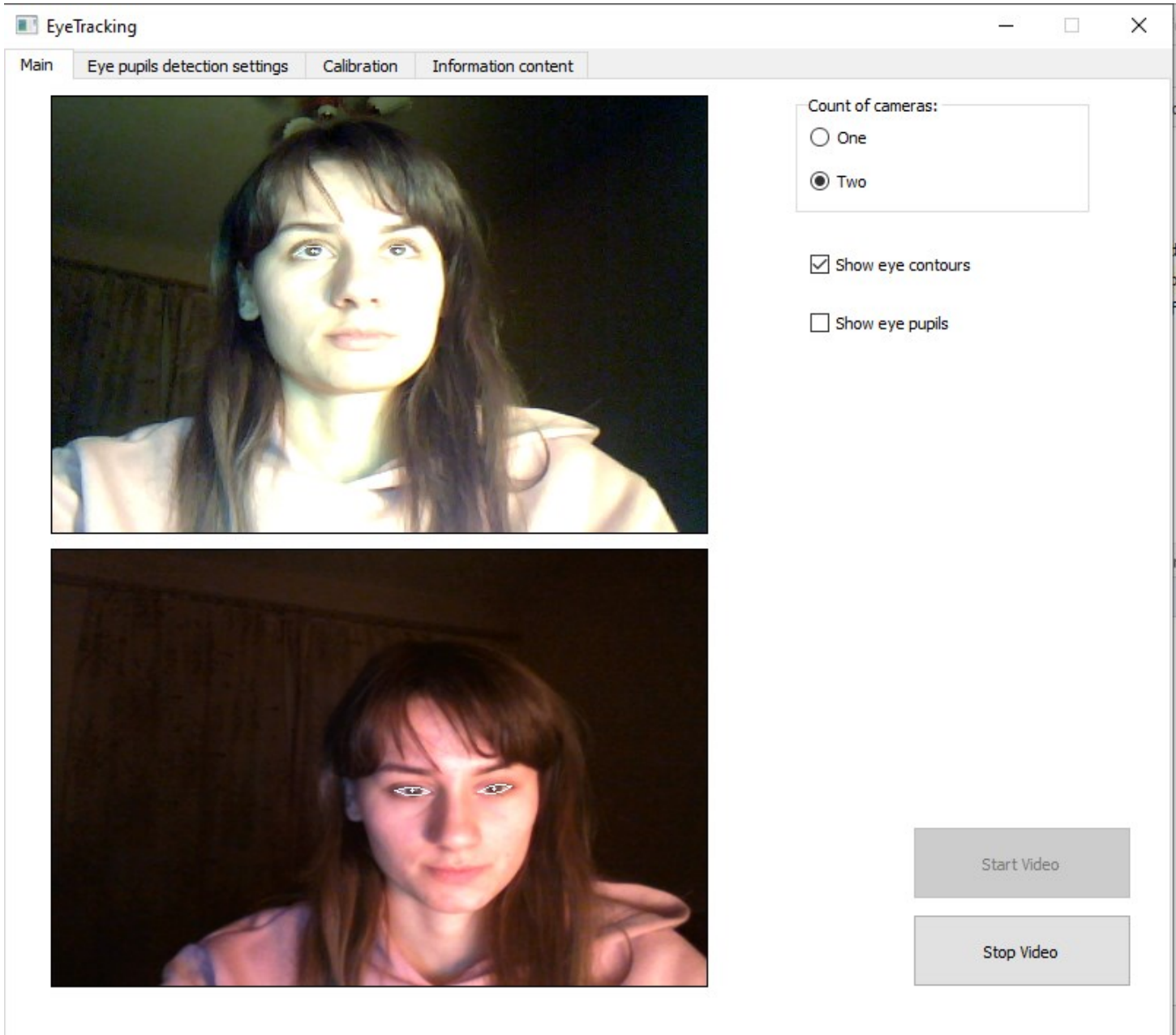


Рисунок 8.1 – Вигляд головного вікна під час обробки відеокadrів з двох камер

Для налаштування калібровки пропонується встановити наступні опції:

- форма калібрувальної точки;
- колір калібрувальної точки (у кольоровій моделі RGB);
- розмір калібрувальної точки (у пікселях);
- проміжок часу зміни позиції калібрувальної точки (у секундах);
- додаткові нотатки до опису налаштування;

- кількість позицій по горизонталі та вертикалі;
- напрямок переміщення позиції калібрувальної точки: по горизонталі, вертикалі, діагоналі;
- назва папки, де буде зберігатися результат калібрування.

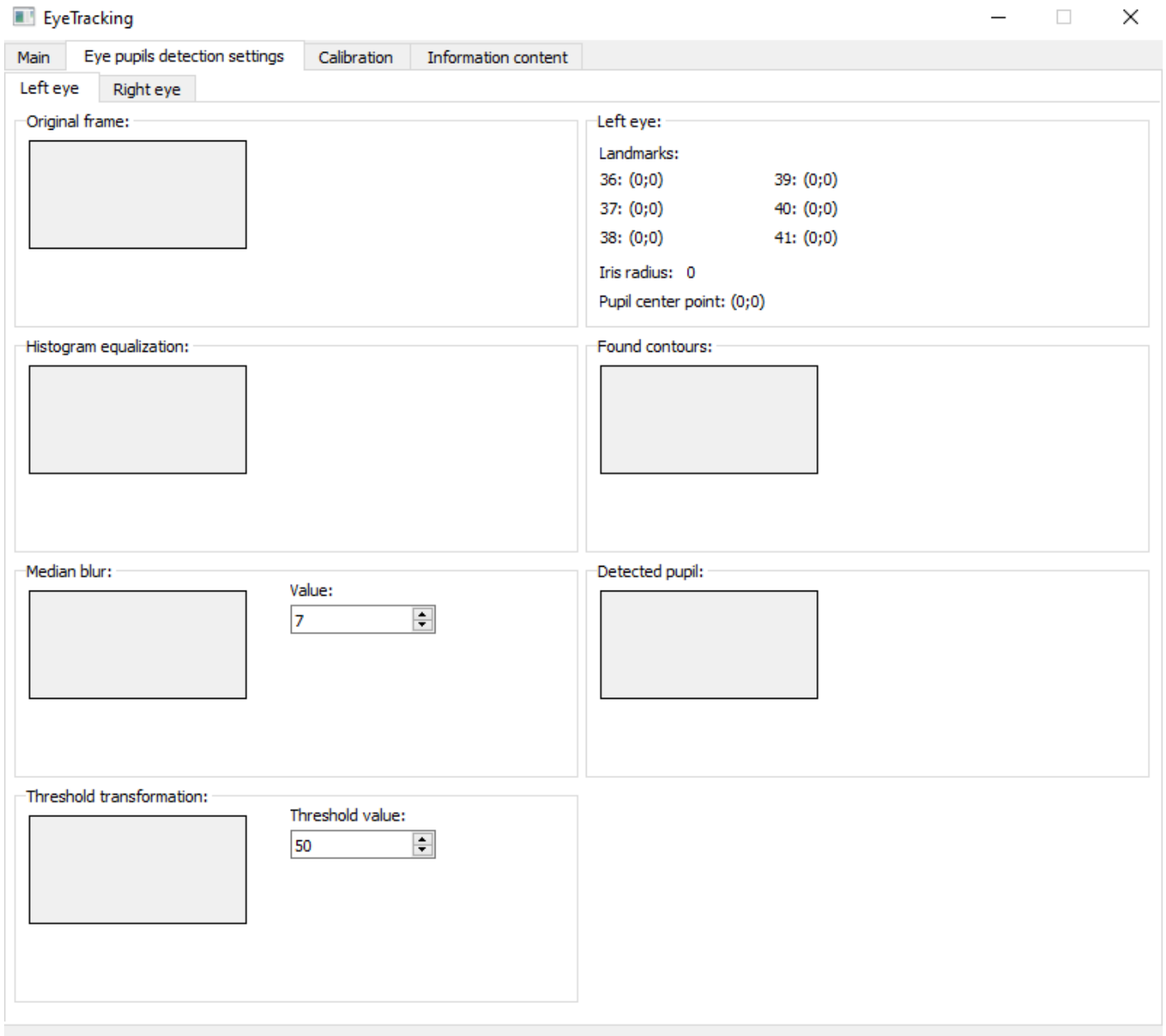


Рисунок 8.2 – Налаштування процесу відстеження зіниці ока

The screenshot shows the 'EyeTracking' application window with the 'Calibration' tab selected. The window has a title bar with standard minimize, maximize, and close buttons. Below the title bar is a tabbed interface with three tabs: 'Main', 'Eye pupils detection settings', and 'Calibration' (which is active). The 'Calibration' tab contains several sections for configuring the calibration process:

- Settings**
 - Form**: Two radio buttons, 'Circle' (selected) and 'Rectangle'.
 - Color**: Three input fields for RGB values: R: 255, G: 0, B: 0. Each field has up and down arrow buttons.
 - Timer**: An input field with the value 5.00 and up/down arrow buttons.
 - Size**: An input field with the value 20 and up/down arrow buttons.
 - Description**: A text area containing the placeholder text 'Description of experiments conditions...'. It has a small 'x' icon in the top right corner.
- Separated points**
 - Capacity**: A label above two input fields.
 - Width**: An input field with the value 3 and up/down arrow buttons.
 - Height**: An input field with the value 3 and up/down arrow buttons.
 - Save**: A button to save the settings.
- Moving point**
 - Step**: An input field with the value 10 and up/down arrow buttons.
 - Direction**: Three radio buttons, 'Vertical' (selected), 'Horizontal', and 'Diagonal'.
 - Save**: A button to save the settings.

At the bottom of the window, there are two more elements:

- A text label 'Enter a folder name for saving calibration result:' followed by an empty text input field.
- A 'Start calibration' button.
- A 'Start testing point of gaze detection' button.

Рисунок 8.3 – Зображення вікна налаштування процесу калібрування

Вікно «Інформаційний контент» (рис. 8.4) дозволяє налаштувати інформаційний контент на екрані та розпочати процес відстежувати відповідності фокусу зору зі структурою інформації на моніторі.

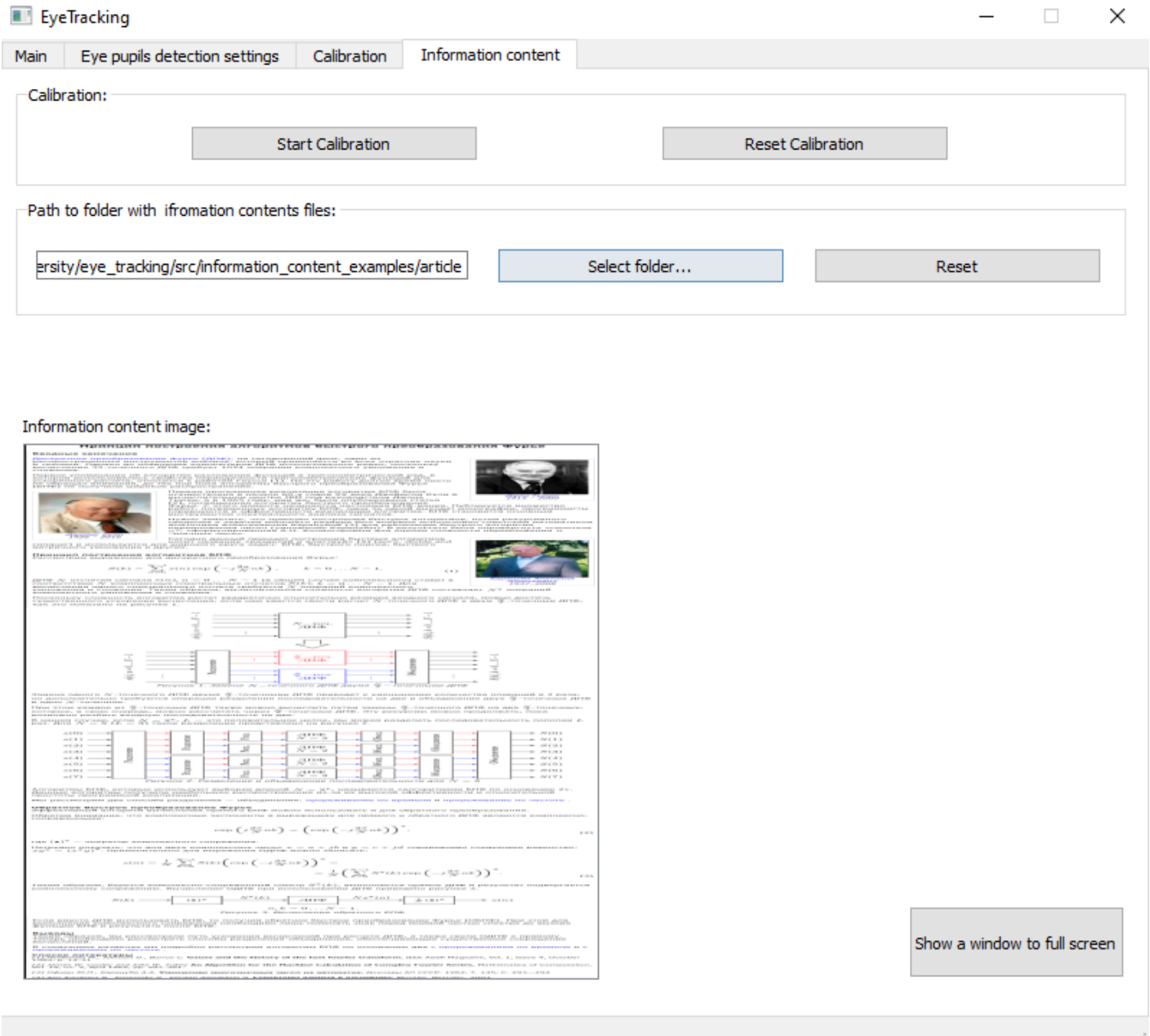


Рисунок 8.4 – Зображення вікна налаштування інформаційного контенту на екрані

9 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

У разі виникнення питань, пов'язаних с продуктом «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера», можна подати запит за електронною адресою: eye_tracking@tracking.com. Підтримка з 9:00 до 18:00 по буднях.

10 ПОВІДОМЛЕННЯ

У табл. 10.1 наведені повідомлення користувачу, що можуть з'явитись у процесі роботи з програмою.

Таблиця 10.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Відсутня камера	Камера не підключена до робочої машини	Перевірте підключення камери до робочої машини
Відсутня можливість отримувати відео з камери	Камера підключена до робочої машини, але використовується іншою програмою	Закрийте програму, яка отримує відео з камери
Обробка даних...	Обробка даних після процесу калібровки	Очікування завершення обробки
Обробка завершена успішно	Обробка даних завершена успішно	Закрийте повідомлення

ЗАТВЕРДЖЕНО

01116130.01192-01 ІЗ 01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ
ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ КОМП'ЮТЕРА

Керівництво користувача. Керівництво відстеження фокусу зору користувача

01116130.01192-01 ІЗ 01

Листів 17

АНОТАЦІЯ

Документ 01116130.01192-01 ІЗ 01 «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера. Керівництво відстеження фокусу зору користувача» входить до складу програмної документації до програми.

Програмний продукт дозволяє розпізнавати фокус зору користувача, шляхом обробки кадрів, отриманих з відеокамери та шукає відповідність отриманих координат фокусу зору зі структурою інформації на моніторі.

ЗМІСТ

Вступ.....	4
1 Призначення та умови застосування.....	5
2 Підготовка до роботи.....	6
3 Опис операцій.....	7
3.1 Налаштування відеокамер.....	7
3.2 Розпізнавання контурів очей та зіниць на відеокадрі	9
3.3 Процес калібрування.....	12
3.4 Налаштування інформаційного контенту.....	13
3.5 Відстеження відповідності отриманих координат фокусу зору зі структурою інформації на моніторі.....	13
4 Аварійні ситуації.....	15
5 Рекомендації щодо засвоєння	16

ВСТУП

Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера дозволяє зробити висновки щодо необхідності редагування інформаційного контенту, для покращення концентрації уваги студента під час дистанційного навчання.

Сфера застосування: навчальні заклади.

Користувачі системи відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера – дослідники, які займаються вивченням поведінки студентів під час роботи з навчальним матеріалом.

Користувачі системи повинні мати досвід роботи з ПК та використання операційної системи Microsoft Windows 10.

Адміністратор системи повинен володіти досвідом встановлення та налаштування програмних та апаратних засобів у платформі Microsoft Windows.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера призначена для відстеження відповідності фокусу зору, задля дослідження поведінки користувача під час роботи з інформаційним контентом. Результат дослідження дозволить зрозуміти, на які елементи інформаційного контенту зосереджено більше уваги, а які залишилися поза увагою, скільки часу знадобилося для засвоєння даного матеріалу, тощо. Дану інформацію можна використовувати для налаштування навчального контенту та можливості адаптувати, налаштовувати та оптимізувати процес навчання.

Умови застосування системи:

- ЕВМ, що має щонайменше 128Мб вільного місця на вбудованому носії інформації та 1024Мб оперативної пам'яті;
- відеокамера з роздільною здатністю 1280x720;
- додаткова зовнішня відеокамера з роздільною здатністю 1280x1024;
- монітор з роздільною здатністю 1366x768;
- маніпулятор «миша» та клавіатуру;
- операційна система Windows.

2 ПІДГОТОВКА ДО РОБОТИ

На дистрибутивному носії даних розміщений виконуваний файл eye_tracking.exe, за допомогою якого можна запустити автоматизовану систему та необхідні для повноцінної роботи зовнішні бібліотеки.

Для запуску програмного продукту «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера» необхідно виконати запуск файлу eye_tracking.exe.

3 ОПИС ОПЕРАЦІЙ

Керівництво відстеження фокусу зору користувача складається з наступних операцій:

- налаштування відеокамер;
- перехват відеокадрів з камер;
- розпізнавання контурів очей та зіниць на відеокадрі;
- процес калібрування;
- налаштування інформаційного контенту;
- відстеження відповідності отриманих координат фокусу зору зі структурою інформації на моніторі.

3.1 Налаштування відеокамер

Налаштуйте відеокамери, які будуть використовуватися для розпізнавання фокусу зору. Розташуйте відеокамери таким чином, що вони мали змогу захоплювати обличчя користувача. Рекомендується використовувати дві камери: відеокамера з роздільною здатністю 1280x720, розташована на висоті 30 см від поверхні столу, в верхній частині, посередині екрану монітора та відеокамера з роздільною здатністю 1280x1024, яка розташована на висоті 10 см від поверхні столу, в нижній частині посередині екрану монітора.

Оберіть кількість відеокамер які будуть використовуватися для розпізнавання фокусу зору (рис. 3.1). Система використовує одну камеру за замовчуванням.

3.2 Перехват відеокадрів з камер

Натисніть «Start Video» (рис. 3.2), щоб система мала змогу перехватувати кадри з відеокамер.

Переконайтесь, що автоматизована система транслює отримані відеокадри у головному вікні (рис. 3.3).

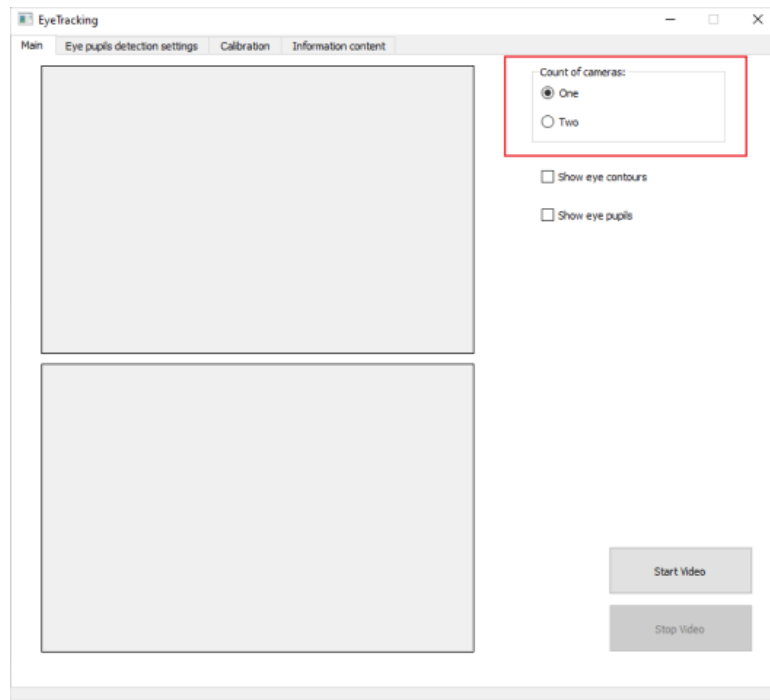


Рисунок 3.1 – Головне вікно програми EyeTracking, з виділеною областю розташування елементів для вибору кількості використання відеокамер

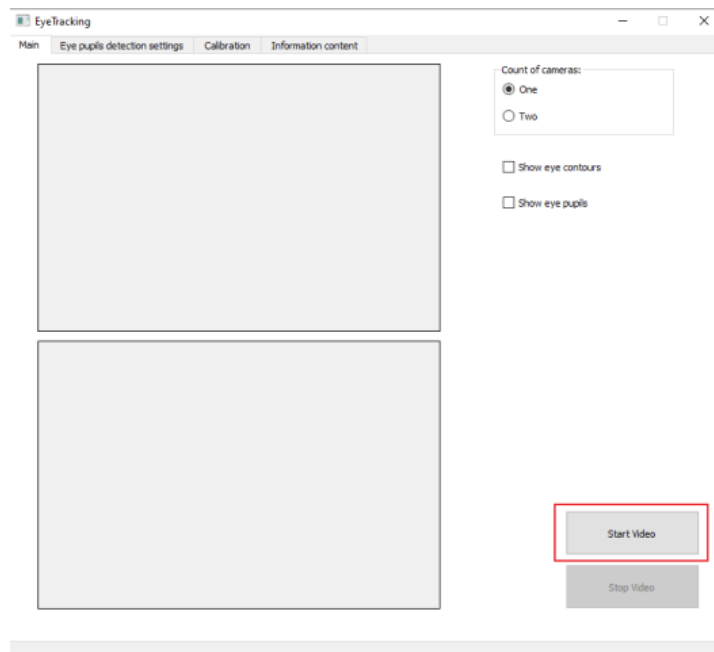


Рисунок 3.2 – Головне вікно програми EyeTracking, з виділеною областю розташування кнопки «Start Video»

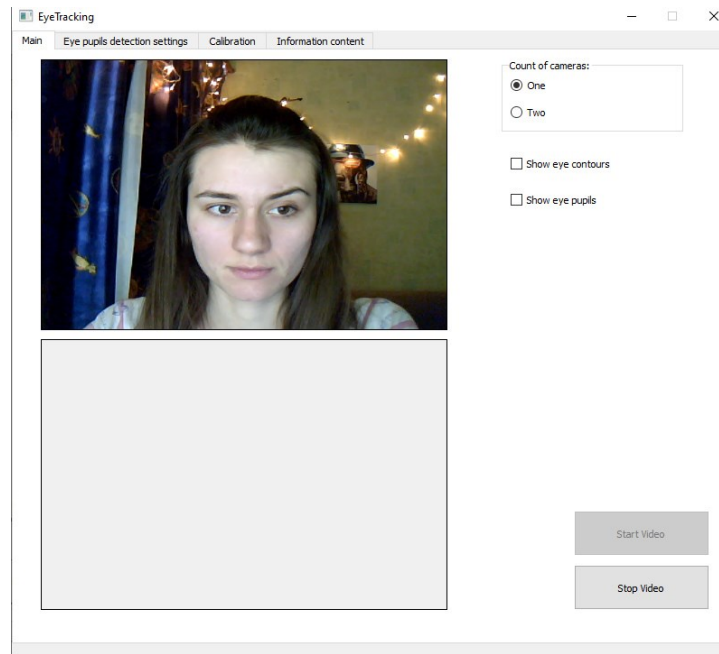


Рисунок 3.3 – Головне вікно програми EyeTracking під час трансляції відеокadrів з відеокамери

3.2 Розпізнавання контурів очей та зіниць на відеокadrі

Переконайтесь, що система обробляє відеокadри, розпізнаючи контури очей та зіниць. Натисніть на елемент «Show eye contours» або «Show eye pupils» (рис. 3.4).

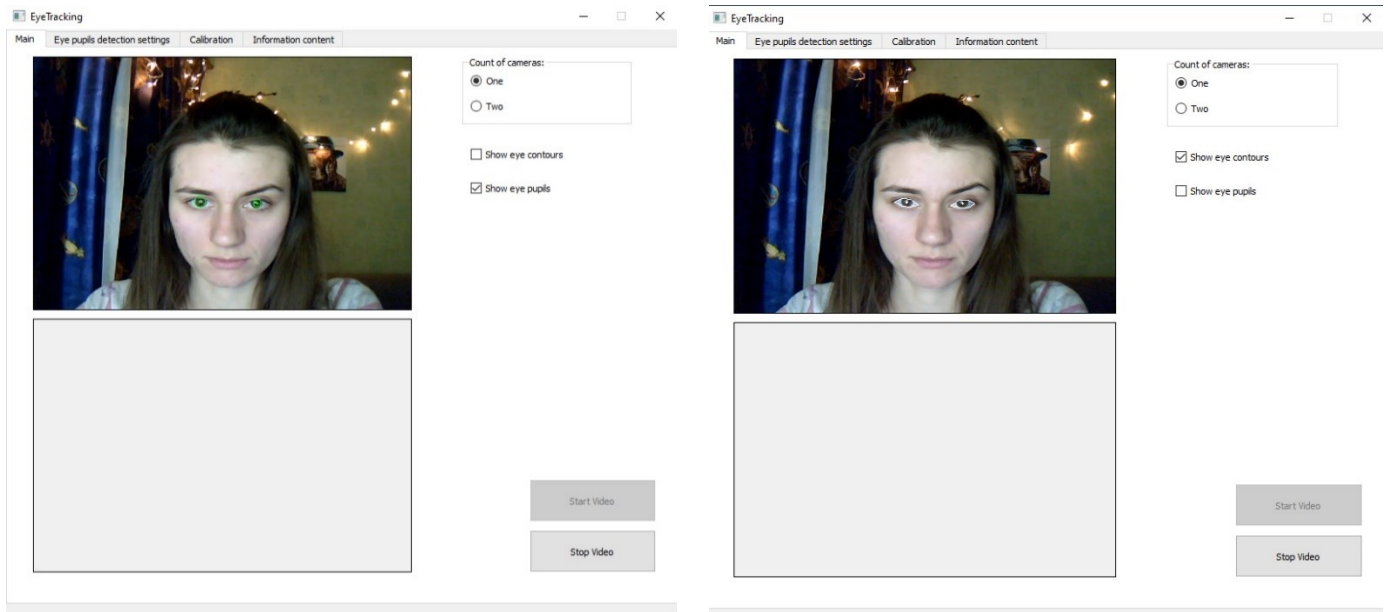


Рисунок 3.4 – Головне вікно під час відстеження контурів очей та зіниць

Щоб налаштувати розпізнавання зіниць очей, натисніть «Eye pupils detection settings» (рис. 3.5). Система відкриє вікно налаштування (рис 3.6), яке дозволяє налаштовувати процес розпізнавання зіниці ока для лівого та правого ока окремо, а також відстежувати обробку кадру під час розпізнавання. Підберіть такі параметри, які б дозволяли найкраще розпізнати область зіниці ока. Встановіть значення апертури медіанного фільтру – непарне натурального числа (рис. 3.7) та порогове значення – у інтервалі натуральних чисел від 0 до 255 (рис. 3.8). Переконайтесь, що встановлені параметри дозволяють розпізнавати область зіниці ока.

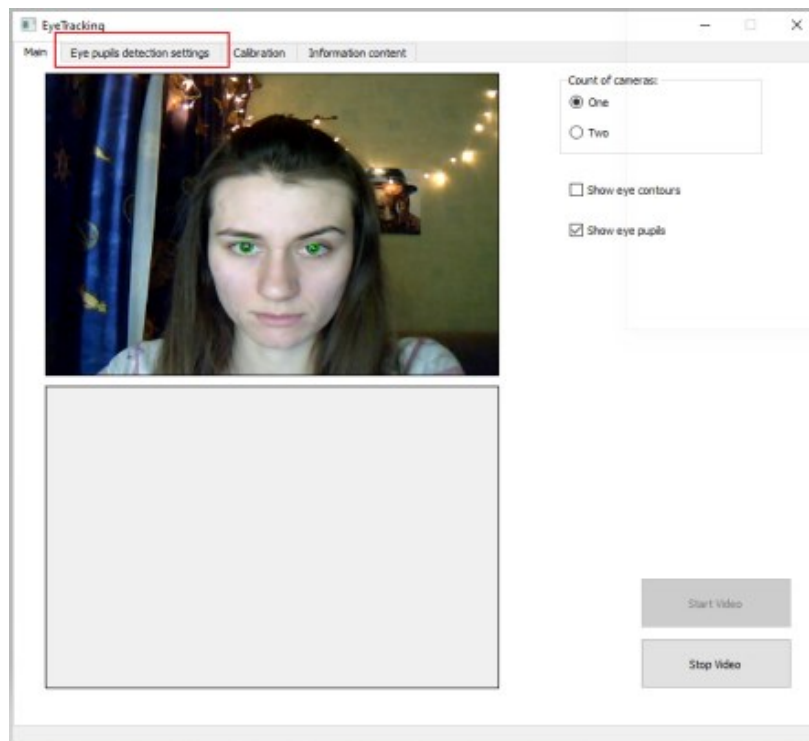


Рисунок 3.5 – Головне вікно програми EyeTracking, з виділеною областю розташування кнопки «Eye pupils detection settings» для переходу у вікно налаштування розпізнавання зіниць очей

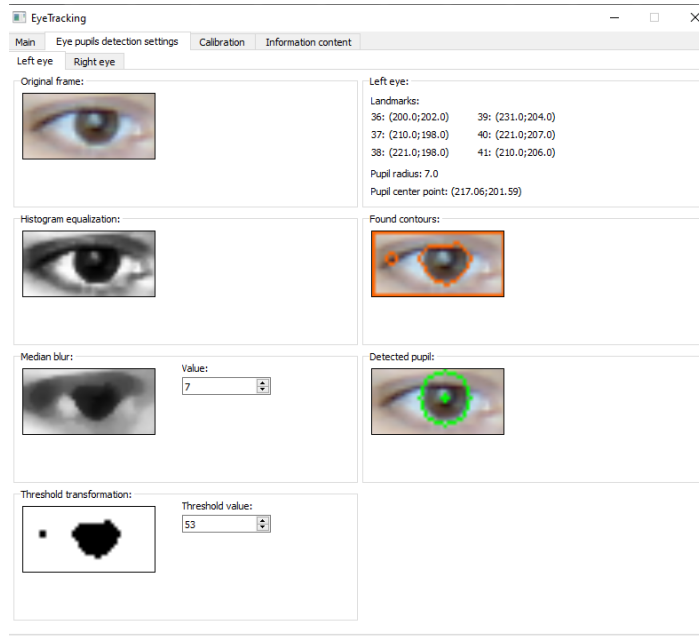


Рисунок 3.6 – Вікно налаштування розпізнавання зіниці лівого ока

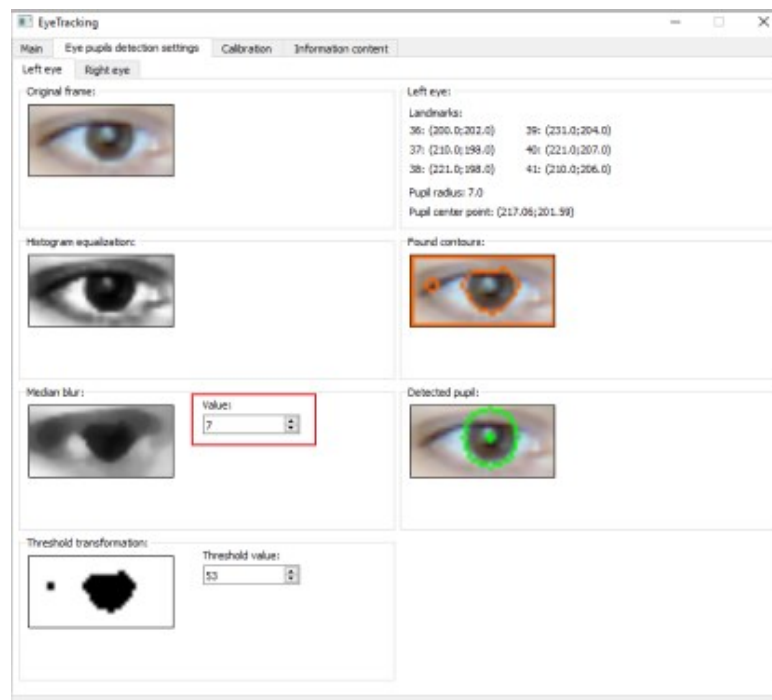


Рисунок 3.7 – Вікно налаштування розпізнавання зіниці лівого ока з виділеною областю встановлення значення апертури медіанного фільтру

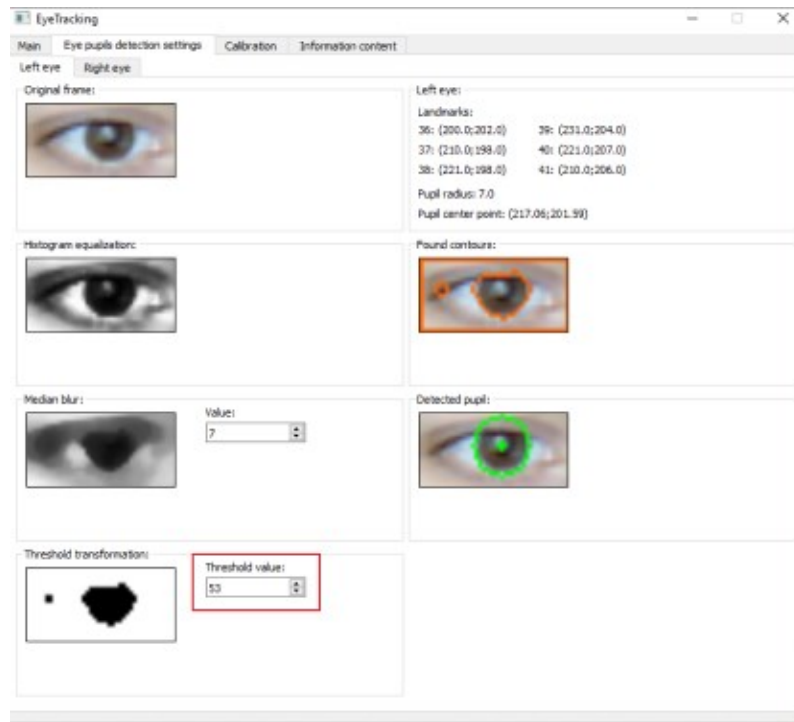


Рисунок 3.8 – Вікно налаштування розпізнавання зіниці лівого ока з виділеною областю встановлення порогового значення

3.3 Процес калібрування

Для покращення відстеження координат фокусу зору, натисніть «Information content», а далі «Start Calibration» (рис. 3.9).

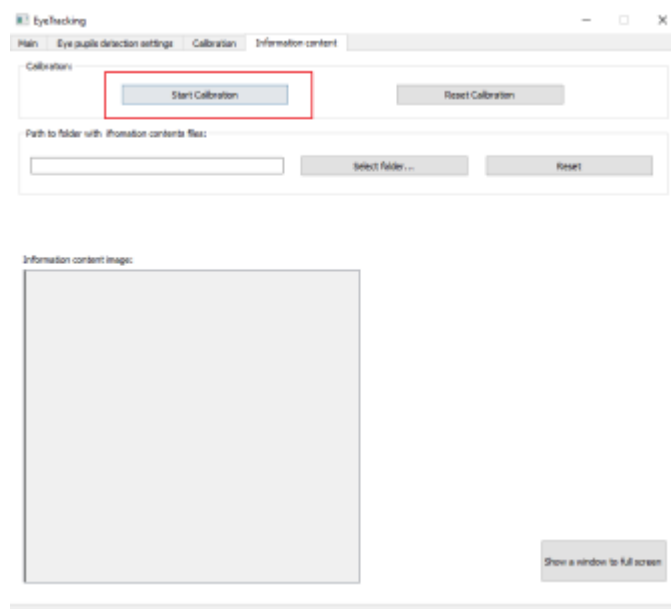


Рисунок 3.9 – Вікно для запуску процесу калібрування

Переконайтесь, що з'явилося вікно на весь екран, де рухається калібрувальна точка, змінюючи свою позицію (рис. 3.10). Слідкуйте за цією точкою. Після завершення процесу калібрування, почекайте, доки система обробить отримані дані. Якщо є потреба видалити результат калібрування, натисніть «Reset Calibration».

3.4 Налаштування інформаційного контенту

Натисніть «Select folder...» та оберіть папку, де знаходиться файл з інформаційний контентом (рис. 3.11), щоб відстежувати відповідність фокусу зору зі інформаційним контентом на моніторі, необхідно налаштувати інформаційний контент. Переконайтеся, що система розпізнала отримані дані.

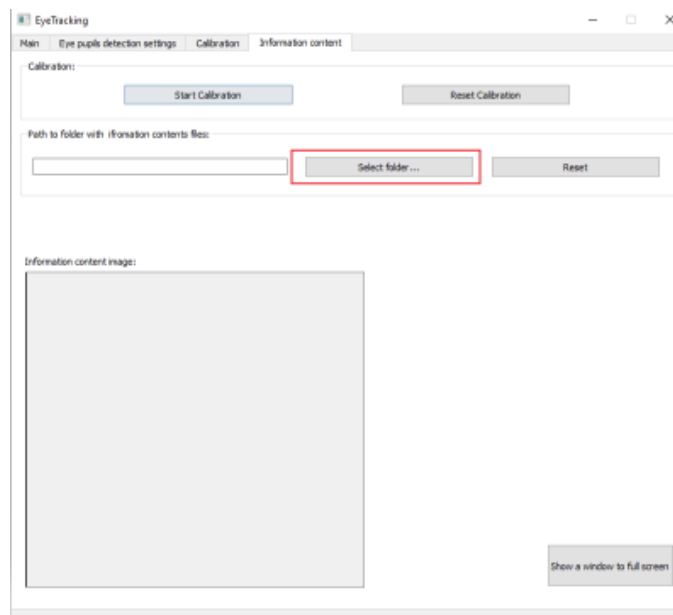


Рисунок 3.11 – Вікно для налаштування інформаційного контенту

3.5 Відстеження відповідності отриманих координат фокусу зору зі структурою інформації на моніторі

Після того, як інформаційний контент буде обрано, натисніть «Show a window to full screen», щоб відобразити інформаційний контент на моніторі екрану та розпочати пошук відповідності отриманих координат фокусу зору зі структурою інформації на моніторі (рис. 3.12).



Рисунок 3.12 – Відображення знаходження відповідності отриманих координат фокусу зору зі структурою інформації на моніторі

Ознайомтесь з відображенням інформаційним контентом. Система розпізнає фокус зору користувача та виділяє червоним пунктиром елемент інформаційного контенту, куди сфокусовано погляд. Також відображається червона точка, яка відповідає координатам фокусу зору.

Знайдена відповідність отриманих координат фокусу зору зі структурою інформації на моніторі записується у файл у наступному вигляді: точка фокусу зору (значення представлене у вигляді двох координат по осі X та Y), ім'я елементу інформаційного контенту, область розташування елементу інформаційного контенту на моніторі (представлена у вигляді координат вершин чотирикутника). Файл створюється біля файлу eye_tracking.exe. Переконайтеся, що

Щоб завершити процес відстеження відповідності натисніть клавішу «s».

4 АВАРІЙНІ СИТУАЦІЇ

Під час експлуатації автоматизованої системи відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера можуть виникати аварійні ситуації. Якщо під час роботи системи буде відсутній зв'язок з відеокамерою, спробуйте перевантажити систему, якщо це не допоможе, зверніться до технічної підтримки.

5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Під час проведення процесу калібрування, намагайтеся не відводити свій погляд від позиції калібрувальної точки, адже це може вплинути на результат розпізнавання координат фокусу зору.

Щоб впевнитися, що система працює коректно, переконайтесь, що система правильно відображає мітку знаходження координат фокусу зору. Мітка представлена у вигляді червоного кола (рис. 3.12).

ЗАТВЕРДЖЕНО

01116130.01192-01 12 01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА ВІДСТЕЖЕННЯ ВІДПОВІДНОСТІ ФОКУСУ
ЗОРУ ЗІ СТРУКТУРОЮ ІНФОРМАЦІЇ НА МОНІТОРІ КОМП'ЮТЕРА

Текст програми

01116130.01192-01 12 01

Листів 47

АНОТАЦІЯ

Документ 01116130.01192-01 12 01 «Автоматизована система відстеження відповідності фокусу зору зі структурою інформації на моніторі. Текст програми» входить до складу програмної документації до програми, яка розпізнає фокус зору користувача, шляхом обробки кадрів, отриманих з відеокамери та шукає відповідність отриманих координат фокусу зору зі структурою інформації на моніторі.

У даному документі представлений текст програм. Програма написана на мові Python. Програма реалізована у програмному середовищі PyCharm. Об'єм пам'яті, що займає програма, складає 40 Мб. Конфігурація комп'ютера: процесор Intel Core i5-3230M, RAM – 4Gb, відеокарта – NVIDIA GeForce GT 740M, монітор з роздільною здатністю 1366x768. Комплекс функціонує в середовищі WINDOWS 10. Додаткові пристрої: відеокамера з роздільною здатністю 1280x720 та додаткова відеокамера з роздільною здатністю 1280x1024.

ЗМІСТ

1	Схема взаємодії програмних модулів	5
2	Текст програми	6
2.1	Пакет FaceDetector	6
2.1.1	Текст програми у файлі eye.py.....	6
2.1.2	Текст програми у файлі eye_detector.py.....	8
2.1.3	Текст програми у файлі pupil.py	9
2.1.4	Зміст файлу pupil_detector.py	10
2.1.5	Текст програми у файлі face.py.....	12
2.1.6	Текст програми у файлі face_detector.py.....	14
2.2	Пакет VideoCapturer.....	15
2.2.1	Текст програми у файлі video_capture.py	15
2.2.2	Текст програми у файлі frame_capturer.py.....	16
2.2.3	Текст програми у файлі frame_analyzer.py	17
2.3	Пакет FaceStabilizer	18
2.3.1	Текст програми у файлі face_stabilizer.py.....	18
2.3.2	Текст програми у файлі eye_stabilizer.py.....	18
2.3.3	Текст програми у файлі pupil_stabilizer.py	20
2.4	Пакет CalibrationModule	20
2.4.1	Текст програми у файлі calibration_module.py	20
2.4.2	Текст програми у файлі calibration_settings.py.....	23
2.4.3	Текст програми у файлі calibration_widget.py	24
2.4.4	Текст програми у файлі data_analyzer.py.....	26

2.4.5 Текст програми у файлі <code>experiment_data.py</code>	27
2.5 Пакет GUI.....	31
2.5.1 Текст програми у файлі <code>main.py</code>	31
2.5.2 Текст програми у файлі <code>calibration_tab.py</code>	34
2.5.3 Текст програми у файлі <code>information_content_tab.py</code>	37
2.5.4 Текст програми у файлі <code>information_content_widget.py</code>	39
2.5.5 Текст програми у файлі <code>left_eye_settings.py</code>	41
2.5.6 Текст програми у файлі <code>right_eye_settings.py</code>	42
2.5.7 Текст програми у файлі <code>eye_settings.py</code>	43
2.6 Пакет <code>PointGazeDetector</code>	45
2.6.1 Текст програми у файлі <code>point_gaze_detector.py</code>	45
2.6.2 Текст програми у файлі <code>point_gaze_handler.py</code>	46

1 СХЕМА ВЗАЄМОДІЇ ПРОГРАМНИХ МОДУЛІВ

На рис. 1.1 представлено схему зображення взаємодії модулів системи відстеження відповідності фокусу зору зі структурою інформації на моніторі

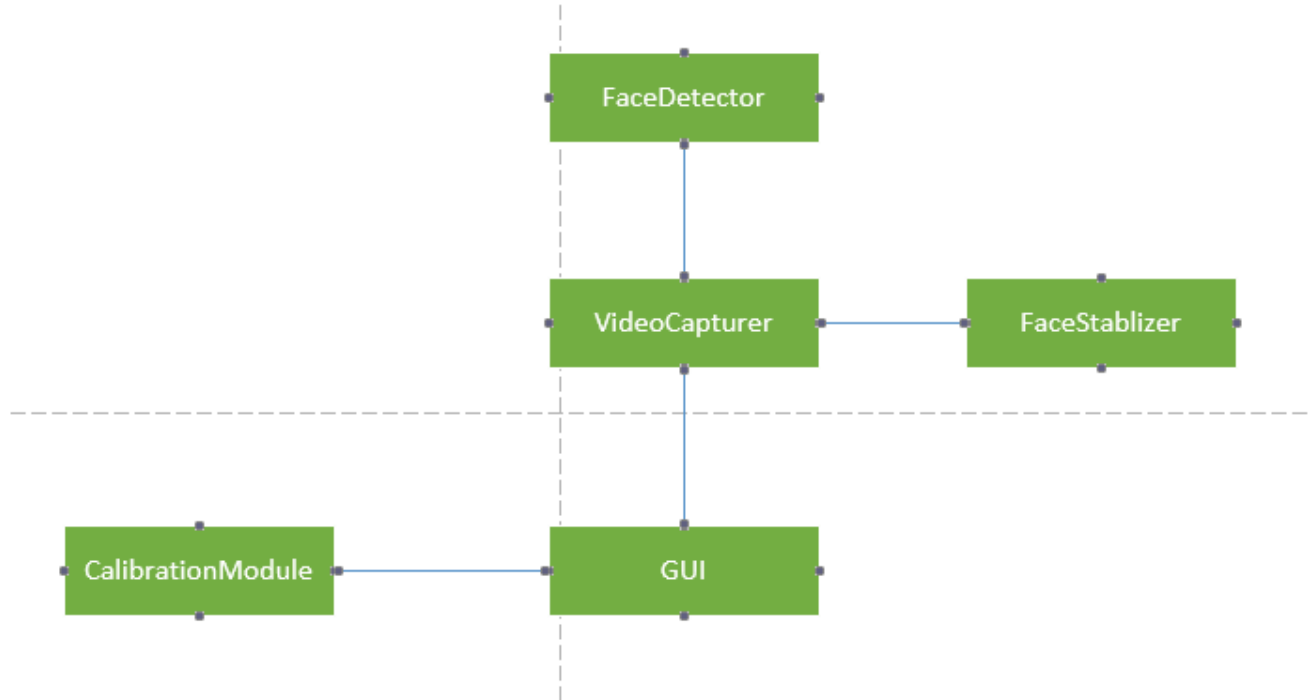


Рисунок 1.1 – Схема взаємодії модулів системи

Пакет **FaceDetector** дозволяє розпізнати образ обличчя на відеокадрі.

Пакет **FaceStablizer** дозволяє стабілізувати отримані дані про розпізнані образи обличчя для кількох кадрів, щоб позбутися шумів під час підрахунку координат фокусу зору.

Пакет **PointGazeDetector** дозволяє відстежити фокус зору під час роботи з інформаційним контентом.

Пакет **VideoCapturer** дозволяє отримувати відеокадри з відеокамер.

Пакет **CalibrationModule** відповідає за процес калібрування.

Пакет **GUI** відповідає за відображення графічного інтерфейсу та обробку сигналів на дії користувача.

2 ТЕКСТ ПРОГРАМИ

2.1 Пакет FaceDetector

2.1.1 Текст програми у файлі eye.py

```
import numpy as np
import cv2
import math
from src.gaze_tracking.eye_pupil.pupil_detector import PupilDetector
from src.gaze_tracking.eye.eye_detector import EyeDetector
from src.gaze_tracking.eye_flare.blick_detector import EyeBlickDetector

class Eye:
    """
    This class creates a new frame to isolate the eye and
    initiates the pupil detection.
    """

    def __init__(self, origin_frame, landmarks, points, params_settings):
        self._eye_detector = EyeDetector(landmarks, points)
        self.size = self._eye_detector.size
        self.pupil = None
        self.frame = None
        self._eye_coord = None
        self._center = None
        self.ear = None
        self._blick = None
        self.filter_frames = []
        self.coe = None

        self._analyze(origin_frame, params_settings)

    def set_data(self, pupil, size, ear, coe):
        self.size = size
        self.pupil = pupil
        self.ear = ear
        self.coe = coe

    def _analyze(self, face_frame, params):
        self.ear = self._eye_detector.ear_aspect_ratio()
        self.coe = self._eye_detector.CoE
        self._eye_coord, self._center, self.frame = self._eye_detector.isolate(face_frame)
        self.pupil, self.filter_frames = self._detect_eye_pupil(params)

        if self.pupil and self.pupil.exist:
            self.filter_frames = self.filter_frames

    def _detect_eye_pupil(self, params):
        pupil_detector = PupilDetector(params)
        pupil, filter_frames = pupil_detector.detect(self.frame)
        if pupil:
```

01116130.01192-01 12 01

7

```
pupil.set_start_point(self._eye_coord)
return pupil, filter_frames

def _detect_eye_blick(self, frame, pupil, threshold, threshold_type):
    blick_detector = EyeBlickDetector(frame, pupil)
    eye_blick, eye_pupil_frame = blick_detector.detect(threshold, threshold_type)
    if eye_blick:
        eye_blick.set_start_point(self._eye_coord)
    return eye_blick, eye_pupil_frame

def set_pupil(self, pupil):
    self.pupil = pupil
    if pupil is None:
        return

def draw_eye_contour(self, frame):
    if self._eye_detector:
        eye_region = self._eye_detector.eye_region()
        frame = cv2.polylines(frame, [eye_region], True, (255, 255, 255), 1)
        frame = cv2.circle(frame, self._eye_detector.CoE, 1, (255, 255, 255), -1)
    return frame

def draw_eye_pupil(self, frame):
    if self.pupil:
        return self.pupil.draw(frame)

@property
def data(self):
    # eye landmarks point, pupil point, iris radius
    if self.exist:
        points = self._eye_detector.data()
        points = points.astype(int)
        points = np.append(points, self.pupil.data())
        return points
    else:
        return None

@property
def calibration_data(self):
    if self.exist:
        points = self._eye_detector.CoE
        points = np.append(points, self.pupil.data())
        return points.astype(int)
    else:
        return None

@property
def exist(self):
    if self.pupil and self.pupil.exist:
```

```

    return True
else:
    return False

```

2.1.2 Текст програми у файлі eye_detector.py

```

import cv2
import numpy as np
import math
from scipy.spatial import distance as dist

class EyeDetector:
    def __init__(self, landmarks, points):
        self._landmarks = landmarks
        self._points = points

    def isolate(self, face_frame):

        region = self.eye_region()

        # Applying a mask to get only the eye
        frame = face_frame
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        height, width = frame.shape[:2]
        black_frame = np.zeros((height, width), np.uint8)

        mask = np.full((height, width), 255, np.uint8)
        cv2.fillPoly(mask, [region], (0, 0, 0))
        eye = cv2.bitwise_not(black_frame, frame.copy(), mask=mask)

        # Cropping on the eye
        margin = 5
        min_x = np.min(region[:, 0]) - margin
        max_x = np.max(region[:, 0]) + margin
        min_y = np.min(region[:, 1]) - margin
        max_y = np.max(region[:, 1]) + margin

        eye_frame = eye[min_y:max_y, min_x:max_x]
        origin_eye_frame = face_frame[min_y:max_y, min_x:max_x]
        eye_origin = (min_x, min_y)

        height, width = eye_frame.shape[:2]
        eye_center = (width / 2, height / 2)

        return eye_origin, eye_center, origin_eye_frame

    def eye_region(self):
        """Return np.array of eye points"""
        region = np.array([(self._landmarks.part(point).x, self._landmarks.part(point).y)
        point in self._points])
        return region.astype(np.int32)

    def data(self):
        points = np.array([])

```

```

    for point in self._points:
        points = np.append(points, [self._landmarks.part(point).x,
self._landmarks.part(point).y])
    return points

def _eye_point(self, index):
    """Get a point using index"""
    return self._landmarks.part(self._points[index])

@property
def CoE(self):
    """Center of eye"""
    eye_region = self.eye_region()
    COEx = int((eye_region[0][0] + eye_region[3][0]) / 2)
    # COEy = int((eye_region[0][1] + eye_region[3][1]) / 2)

    y1 = (eye_region[1][1] + eye_region[2][1]) / 2
    y2 = (eye_region[4][1] + eye_region[5][1]) / 2
    COEy = int((y1 + y2) / 2)

    return (COEx, COEy)

@property
def size(self):
    """Return eye width and height"""
    eye_region = self.eye_region()
    EyeWidth = math.fabs(eye_region[0][0] - eye_region[3][0])

    y1 = (eye_region[1][1] + eye_region[2][1]) / 2
    y2 = (eye_region[4][1] + eye_region[5][1]) / 2
    EyeHeight = math.fabs(y1 - y2)

    return (EyeWidth, EyeHeight)

def eye_aspect_ratio(self):
    eye_region = self.eye_region()

    # compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    A = dist.euclidean(eye_region[1], eye_region[5])
    B = dist.euclidean(eye_region[2], eye_region[4])

    # compute the euclidean distance between the horizontal
    # eye landmark (x, y)-coordinates
    C = dist.euclidean(eye_region[0], eye_region[3])

    # compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)

    # return the eye aspect ratio
    return ear

```

2.1.3 Текст програми у файлі rupil.py

```

import cv2
import numpy as np

```



```

class Pupil:
    def __init__(self, center_point, radius):
        self.center_point = center_point
        self.radius = radius
        self.start_point = (0, 0)

    @property
    def exist(self):
        try:
            self.center_point
            self.radius
            return True
        except Exception:
            return False

    def draw(self, frame):
        if self.exist:
            point = (int(self.center_eye_pupil[0]), int(self.center_eye_pupil[1]))
            frame = cv2.circle(frame, point, 1, (0, 255, 0), -1)
            frame = cv2.circle(frame, point, self.radius, (0, 255, 0), 1)
            return frame

    @property
    def center_eye_pupil(self):
        if not self.exist:
            return None

        x = self.start_point[0] + self.center_point[0]
        y = self.start_point[1] + self.center_point[1]
        return (x, y)

    def data(self):
        if self.exist:
            points = self.center_eye_pupil
            points = np.append(points, self.radius)
            return points
        else:
            return None

    def set_start_point(self, point):
        self.start_point = point

```

2.1.4 Зміст файлу pupil_detector.py

```

import cv2
import numpy as np

from src.gaze_tracking.eye_pupil.pupil import Pupil
from src.utils.utils import Utils

class PupilDetector:

```

```

def __init__(self, params_settings):
    self.threshold = params_settings[0]
    self.thresholding_type = params_settings[1]
    self.medianValue = params_settings[2]

def detect(self, frame):
    frames = self._process(frame)
    frames = [frame] + frames
    pupil, frames = self._detect_pupil(frames)
    return pupil, frames

def _process(self, frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.equalizeHist(gray, gray)
    medianBlur = cv2.medianBlur(gray, self.medianValue)
    thresh = cv2.threshold(medianBlur, self.threshold, 255, self.thresholding_type)[1]
    return [gray, medianBlur, thresh]

def _detect_pupil(self, frames):
    frame = frames[-1]
    contours, _ = cv2.findContours(frame, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    contours = sorted(contours, key=cv2.contourArea)[:10]

    contours_frame = np.copy(frames[0])
    cv2.drawContours(contours_frame, contours, -1, (0, 100, 255), 1)

    try:
        # contours[-1] is a border of frame
        (x, y), radius = cv2.minEnclosingCircle(contours[-2])

        # calculate moments of binary image
        M = cv2.moments(contours[-2])

        # calculate x,y coordinate of center
        cX = M["m10"] / M["m00"]
        cY = M["m01"] / M["m00"]

        pupil = Pupil((Utils.toFixed(cX), Utils.toFixed(cY)), int(radius))

        pupil_frame = np.copy(frames[0])
        pupil_frame = pupil.draw(pupil_frame)

        frames = frames + [contours_frame, pupil_frame]

        return pupil, frames

    except IndexError:
        # print("[Pupil]: Failed to get circle contour")
        return None, []

    except ZeroDivisionError:

```

01116130.01192-01 12 01

12

```
# print("[Pupil]: Failed to get circle contour")
return None, []
```

2.1.5 Текст програми у файлі face.py

```
import numpy as np
import cv2
from src.gaze_tracking.head_side import HeadSide, Side

EYE_AR_THRESH = 0.15

class Face:

    def __init__(self, frame, landmarks, delegate, cameraID):
        self._frame = frame
        self._landmarks = landmarks
        self._delegate = delegate

        self._left_side = HeadSide(frame, landmarks, Side.Left,
        delegate.params_settings(Side.Left), delegate, cameraID)
        self._right_side = HeadSide(frame, landmarks, Side.Right,
        delegate.params_settings(Side.Right), delegate, cameraID)

        self._correct = self._check_side_detection and (not self._is_blinking())

    def show(self, frame):
        frame = self._draw(frame)
        self._display_data()
        return frame

    @property
    def _check_side_detection(self):
        if self._left_side.exist and self._right_side.exist:
            return True
        else:
            return False

    def _is_blinking(self):
        #TODO: add custom constant
        if not self._left_side.eye.ear or not self._right_side.eye.ear:
            return False
        else:
            # average the eye aspect ratio together for both eyes
            ear = (self._left_side.eye.ear + self._right_side.eye.ear) / 2.0
            # print("[EAR]", ear)
            return ear < EYE_AR_THRESH

    def _draw(self, frame):
        if not self._is_blinking():
            frame = self._left_side.draw(frame)
            frame = self._right_side.draw(frame)
```

01116130.01192-01 12 01

13

```
    if self._delegate.does_show_head_area:
        polylines = np.array([(self._landmarks.part(27).x,
self._landmarks.part(27).y),
                               (self._landmarks.part(28).x, self._landmarks.part(28).y),
                               (self._landmarks.part(29).x, self._landmarks.part(29).y)])

        frame = cv2.polylines(frame, [polylines], True, (0, 255, 255), 1)

    return frame

def _display_data(self):
    if self._correct and self._left_side.data is not None and self._right_side.data is
not None:
        self._left_side.display_data()
        self._right_side.display_data()

def save_to_file(self, point):
    if self._delegate.pog_calibration == True:
        self.save_calibration_data_to_file(point)
        return

    data = self.data(point)
    if data is not None:
        self._delegate.save_data_to_file(data)

def data(self):
    if self._correct and self._left_side.data is not None and self._right_side.data is
not None:
        all_points = self._left_side.data
        all_points = np.append(all_points, self._right_side.data)
        all_points = np.append(all_points, (self._landmarks.part(27).x,
self._landmarks.part(27).y))
        all_points = np.append(all_points, (self._landmarks.part(28).x,
self._landmarks.part(28).y))
        all_points = np.append(all_points, (self._landmarks.part(29).x,
self._landmarks.part(29).y))
        return all_points
    return None

def save_calibration_data_to_file(self, point):
    if self._correct and self.left_eye.calibration_data is not None and
self.right_eye.calibration_data is not None:
        all_points = np.array(point)
        all_points = np.append(all_points, self.left_eye.calibration_data)
        all_points = np.append(all_points, self.right_eye.calibration_data)
        self._delegate.save_data_to_file(all_points)

@property
def left_eye(self):
    return self._left_side.eye
```

```

@property
def right_eye(self):
    return self._right_side.eye

def set_eyes(self, l_eye, r_eye):
    self._left_side.eye = l_eye
    self._right_side.eye = r_eye

```

2.1.6 Текст програми у файлі face_detector.py

```

import cv2
import dlib
from ..face.face import Face

faces_detector = dlib.get_frontal_face_detector()
faces_predictor = dlib.shape_predictor("resources/shape_predictor_68_face_landmarks.dat")

class FaceDetector:

    def __init__(self, window_delegate, cameraID):
        self._delegate = window_delegate
        self._cameraID = cameraID
        self.face = None

    def detect(self, frame):
        """Analyze the current frame and detect all faces"""

        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        cv2.equalizeHist(gray_frame, gray_frame)

        faces = self._detect_faces(gray_frame)
        for face in faces:
            landmarks = self._face_landmarks(gray_frame, face)
            return Face(frame, landmarks, self._delegate, self._cameraID)

    def _detect_faces(self, frame):
        """Return array of faces
        Arguments:
            frame - the current frame for analyzing
        """
        return faces_detector(frame)

    def _face_landmarks(self, frame, face):
        """Return all face landmarks
        Arguments:
            face - the face for detecting landmarks
        """
        return faces_predictor(frame, face)

```

2.2 Пакет VideoCapturer

2.2.1 Текст програми у файлі video_capture.py

```
import cv2
from PyQt5.QtCore import QTimer
import datetime

from src.video.frame.frame_analyzer import FrameAnalyzer
from ..utils.utils import Utils

class VideoCapture:

    def __init__(self, number, delegate):
        self._number = number
        self._analyzer = FrameAnalyzer(delegate, number)
        self._delegate = delegate
        self._cap = self.__init_video_capture(number)
        self._out = self.__init_out()
        self._capture_frame = False
        self.start_time = datetime.datetime.now()

        self.timer = QTimer(self._delegate)
        self.timer.timeout.connect(self._update_frame)

        self.count = 0

    def __del__(self):
        if self._cap.isOpened():
            self._cap.release()
        if self._out.isOpened():
            self._out.release()

    def __init_video_capture(self, number):
        cap = cv2.VideoCapture(number)
        if cap.isOpened() is False:
            print('Failed to get video capture from camera #', number)
            raise Exception('Failed to get video capture from camera #', number)
        return cap

    def __init_out(self):
        # Default resolutions of the frame are obtained. The default resolutions are system dependent.
        # We convert the resolutions from float to integer.
        frame_width = int(self._cap.get(3))
        frame_height = int(self._cap.get(4))

        Utils.create_dir('output')
        self.video_name = "output/outpy_" + str(self._number) + ".avi"

        # Define the codec and create VideoWriter object. The output is stored in 'outpy.avi' file.
        return cv2.VideoWriter(self.video_name, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
```

```
10, (frame_width, frame_height))
```

```
def start(self):
    self.start_time = datetime.datetime.now()
    self.count = 0
    self._capture_frame = True
    self.timer.start()

def stop(self):
    self._capture_frame = False
    diff = datetime.datetime.now() - self.start_time
    self.timer.stop()
    # print('[FREQUENCY]: ', diff / self.count)

def _update_frame(self):
    # Capture frame-by-frame
    _, frame = self._cap.read()
    frame = cv2.resize(frame, self._delegate.frame_size())
    if frame is None:
        raise Exception("[VideoCapture] Failed to get frame from camera #", self._number)

    frame = self._analyzer.analyze(frame)
    if frame is not None:
        self._delegate.show_video_frame(self._number, frame)

    self.count += 1

    # Write the frame into the file 'output.avi'
    self._out.write(frame)
```

2.2.2 Текст програми у файлі frame_capturer.py

```
import cv2
import threading

from src.video.video_capture import VideoCapture

class CameraThread(threading.Thread):
    def __init__(self, camID, delegate):
        threading.Thread.__init__(self)
        self.camID = camID
        self.videoCapture = VideoCapture(camID, delegate)

    def run(self):
        print("[CameraThread] Starting capturing video from camera #", self.camID)
        self.videoCapture.start()

    def terminate(self):
        self.videoCapture.stop()

class FrameCapture:
```

```

def __init__(self, cameras_count, delegate):
    self.cameras_count = cameras_count
    self.videoCaptureTreadsArr = []

    self.__initObjs(delegate)

def __initObjs(self, delegate):
    for i in range(self.cameras_count):
        self.videoCaptureTreadsArr.append(VideoCapture(i, delegate))

def start(self):
    for i in range(self.cameras_count):
        self.videoCaptureTreadsArr[i].start()

def stop(self):
    for i in range(self.cameras_count):
        self.videoCaptureTreadsArr[i].stop()
    self.videoCaptureTreadsArr = []

```

2.2.3 Текст програми у файлі frame_analyzer.py

```

import numpy as np
from src.gaze_tracking.face.face_detector import FaceDetector
from src.gaze_tracking.face.face_stabilizer import FaceStabilizer

class FrameAnalyzer:
    """This class analyze a frame to detect faces, eyes"""

    nFrames = 4

    def __init__(self, window_delegate, cameraID):
        self._delegate = window_delegate
        self._face_detector = FaceDetector(self._delegate, cameraID)
        self._faceStabilizer = FaceStabilizer(FrameAnalyzer.nFrames)
        self._faces = []
        self._frames = []
        self._counter = 0
        self._cameraID = cameraID

    def analyze(self, frame):
        """Analyze the current frame"""
        point = self._delegate.calibrationModule.follow_point()

        if self._counter == FrameAnalyzer.nFrames:
            stabilizeFace = self._faceStabilizer.stabilize(self._faces)

            frame = self._frames[int(FrameAnalyzer.nFrames / 2)]
            frame = stabilizeFace.show(frame)

            if stabilizeFace.data() is not None:
                data = point

```



```

        data = np.append(data, stabilizeFace.data())
        self._delegate.processData(data, self._cameraID)
    else:
        print("[FrameAnalyzer]: Data is None")

    self._faces = []
    self._frames = []
    self._counter = 0
    return frame

    face = self._face_detector.detect(frame)
    if face is None:
        return None

    self._faces.append(face)
    self._frames.append(frame)
    self._counter = self._counter + 1
    return None

```

2.3 Пакет FaceStabilizer

2.3.1 Текст програми у файлі face_stabilizer.py

```

from src.gaze_tracking.eye.eye_stabilizer import EyeStabilizer

class FaceStabilizer:

    def __init__(self, nCounter):
        self.nCounter = nCounter

    def stabilize(self, faces):
        left_eyes = []
        right_eyes = []

        for face in faces:
            left_eyes.append(face.left_eye)
            right_eyes.append(face.right_eye)

        n = int(self.nCounter / 2)
        left_eye = EyeStabilizer.stabilize(left_eyes, left_eyes[n])
        right_eye = EyeStabilizer.stabilize(right_eyes, right_eyes[n])

        face = faces[n]
        face.set_eyes(left_eye, right_eye)

        return face

```

2.3.2 Текст програми у файлі eye_stabilizer.py

```

from src.gaze_tracking.eye_pupil.pupil_stabilizer import PupilStabilizer

class EyeStabilizer:

    @staticmethod
    def stabilize(eyes, output_eye):

```

```
stab_size = EyeStabilizer.stabilize_size(eyes)
stab_pupil = PupilStabilizer.stabilize(eyes)
stab_ear = EyeStabilizer.stabilize_ear(eyes)
stab_coe = EyeStabilizer.stabilize_coe(eyes)

output_eye.set_data(stab_pupil, stab_size, stab_ear, stab_coe)
return output_eye
```

```
def stabilize_size(eyes):
    sumW = 0
    sumH = 0
    counter = 0

    for eye in eyes:
        if eye.size:
            w, h = eye.size
            sumW += w
            sumH += h
            counter += 1

    if counter == 0:
        return (0, 0)

    return (int(sumW / counter), int(sumH / counter))
```

```
def stabilize_ear(eyes):
    sum = 0
    counter = 0

    for eye in eyes:
        if eye.ear:
            sum += eye.ear
            counter += 1

    if counter == 0:
        return 0

    return (sum / counter)
```

```
def stabilize_coe(eyes):
    sumX = 0
    sumY = 0
    counter = 0

    for eye in eyes:
        if eye.coe:
            x, y = eye.coe
            sumX += x
            sumY += y
            counter += 1

    if counter == 0:
        return (0, 0)
```

```
return (int(sumX / counter), int(sumY / counter))
```

2.3.3 Текст програми у файлі pupil_stabilizer.py

```
from src.gaze_tracking.eye_pupil.pupil import Pupil
from src.utils.utils import Utils

class PupilStabilizer:

    @staticmethod
    def stabilize(eyes):
        return PupilStabilizer._stabilize_pupil(eyes)

    def _stabilize_pupil(eyes):
        sum_x = 0
        sum_y = 0
        sum_r = 0

        num = 0

        for eye in eyes:
            if eye.pupil is not None and eye.pupil is not None:
                (x, y) = eye.pupil.center_eye_pupil
                sum_x = sum_x + x
                sum_y = sum_y + y
                sum_r = sum_r + eye.pupil.radius
                num = num + 1

        if num == 0:
            return None

        aver_x = Utils.toFixed(sum_x / num)
        aver_y = Utils.toFixed(sum_y / num)
        aver_r = int(sum_r / num)

        return Pupil((aver_x, aver_y), aver_r)
```

2.4 Пакет CalibrationModule

2.4.1 Текст програми у файлі calibration_module.py

```
import cv2
import datetime, time
import threading
import enum
import numpy as np

from .calibration_settings import CalibrationSettings
from ..gui.calibration_widget import CalibrationWidget
from ..calibraion.data_analyzer.data_analyzer import DataAnalyzer
from ..utils.timer import TimerUtils

class CalibrationModule:

    kSkipPointNumber = 2
```

```
class Mode(enum.Enum):
    Calibration = 0
    PointOfGaze = 1

def __init__(self, delegate):
    self.delegate = delegate
    self._settings = None
    self.isSavedDataToFile = False
    self.isDetectedPointGaze = False
    self._expr_dir = None
    self.take_screenshots = False
    self.lock = threading.Lock()
    self.pointGazeDetector = None
    self.calibrationWidget = CalibrationWidget(self)
    self.mode = CalibrationModule.Mode.Calibration
    self.timer = TimerUtils.timer(delegate, self.changePoint)
    self.point = (-1, -1)
    self.index = -1
    self.skipPointNumber = 0

def follow_point(self):
    self.lock.acquire()
    point = self.point
    self.lock.release()
    return point

def start(self, jsonFilePath, dir_name, csvFiles):
    print('[CalibrationModule]: start')
    self.index = -1
    self.point = (-1, -1)
    self._settings = CalibrationSettings(jsonFilePath)
    self._expr_dir = dir_name
    self._csvFiles = csvFiles
    self.flagFinish = False

    self._startShowingCalibrationWindow()
    self.timer.start(self._settings.timer)

def _startShowingCalibrationWindow(self):
    self.calibrationWidget.setPointObj(0, self._settings.point)
    if self.pointGazeDetector:
        self.calibrationWidget.setPointObj(1, self.pointGazeDetector.monitorPointGaze)

    self.calibrationWidget.c_show(self._expr_dir, self.isDetectedPointGaze)

def finishCalibration(self):
    self.stop()

    if self.Mode == CalibrationModule.Mode.Calibration and self.flagFinish == True:
        print("[CalibrationModule]: Start processing calibration data")
        dataAnalyzer = DataAnalyzer(self._expr_dir)
```

```

dataAnalyzer.processCalibrationDataFromCSVFiles(self._csvFiles)
print("[CalibrationModule]: Finish processing calibration data")

def changePointN(self):
    self.changePoint()
    self.timer.stop()
    self.timer.start(self._settings.timer)

def changePoint(self):
    if self.index + 1 >= len(self._settings.coords):
        self.flagFinish = True
        self.finishCalibration()
        return

    self.lock.acquire()
    self.index += 1
    self.point = self._settings.coords[self.index]
    self.skipPointNumber = 0
    print('[New point]: ', self.point, datetime.datetime.today())
    self.calibrationWidget.draw(self.point, 0)
    self.lock.release()

def stop(self):
    print('[CalibrationModule]: stop')
    self.isSavedDataToFile = False
    self.isDetectedPointGaze = False
    self.index = -1
    self.calibrationWidget.c_close()
    self.timer.stop()

def save_data_to_file(self, data, csv_file):
    if self.isSavedDataToFile and data[0] != -1:
        print('[Data] : ', csv_file.file_name, data)
        data = data.astype(float, 2)
        csv_file.write_data(data)
        if self.take_screenshots:
            self.delegate.take_snapshots(self._expr_dir, csv_file.row_index)
            self.take_screenshots = False

def detectPointGaze(self, pairDataDict):
    self.lock.acquire()

    if self.skipPointNumber != CalibrationModule.kSkipPointNumber:
        self.skipPointNumber += 1
        self.calibrationWidget.draw((-1, -1), 1)
    elif self.point[0] != -1:
        point = self.pointGazeDetector.detectPointGaze(pairDataDict)
        print("[CalibrationModule]: Monitor coordinates: ", point,
datetime.datetime.today())
        self.calibrationWidget.draw(point, 1)

```

```

# save to file
data = np.array(self.point)
data = np.append(data, point)
self.pointGazeDetector.csvFile.write_data(data)

self.lock.release()

```

2.4.2 Текст програми у файлі calibration_settings.py

```

import json

from ..constants.constants import Const
from src.point_gaze.point_gaze import GazePoint

class CalibrationSettings:
    def __init__(self, filePath):
        self.point = None
        self.timer = None
        self.coords = None
        self.delim = Const.k_csv_delim

        self._load_json(filePath)

    def _load_json(self, filePath):
        try:
            with open(filePath, "r", encoding='utf-8') as read_file:
                json_data = read_file.read()
                data = json.loads(json_data)

                self.point = self._point(data)
                self.timer = self._timer(data)
                self.delim = self._delim(data)
                self.coords = self._coords(data)
        except FileNotFoundError:
            print('No such file or directory: resources/calibration_settings.json')

    def _point(self, data):
        point = data[Const.k_point_key]
        return GazePoint(point[Const.k_point_form_key],
                          point[Const.k_point_size_key],
                          self._color_tuple(point))

    def _color_tuple(self, data):
        arr = data[Const.k_point_color_key]
        return (arr[0], arr[1], arr[2])

    def _timer(self, data):
        return (data[Const.k_timer_key] * 1000)

    def _delim(self, data):
        return data[Const.k_delim_key]

```

```

def _coords(self, data):
    input_coords = data[Const.k_coordinates_key]
    output_coords = []

    for point in input_coords:
        output_coords.append((point[0], point[1]))

    return output_coords

```

2.4.3 Текст програми у файлі calibration_widget.py

```

from PyQt5.QtWidgets import (QWidget)
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QPainter, QColor
import threading
import math

from ..video.screen_recorder import ScreenRecorder

class CalibrationWidget(QWidget):

    def __init__(self, delegate):
        super().__init__()
        self.initUI()
        self.pointDict = dict()
        self._delegate = delegate
        self.screenRecorder = ScreenRecorder(self)
        self.lock = threading.Lock()
        self.startRecordingFlag = False
        self.error = (0, 0)
        self.averError = (0, 0)
        self.countError = 0
        self.sumError = (0, 0)

    def initUI(self):
        self.setCursor(Qt.BlankCursor)
        self.setBackground(QColor(153, 153, 153))
        self.qp = QPainter()

    def setBackground(self, qColor):
        p = self.palette()
        p.setColor(self.backgroundRole(), qColor)
        self.setPalette(p)

    def c_show(self, dirPath, flag):
        self.startRecordingFlag = flag
        self.setBackground(QColor(153, 153, 153))
        self.setCursor(Qt.BlankCursor)
        self.showFullScreen()
        if self.startRecordingFlag:
            self.screenRecorder.start(dirPath)

```

```

def c_close(self):
    self.qp.restore()
    if self.startRecordingFlag:
        self.screenRecorder.stop()
    self.setCursor(Qt.ArrowCursor)
    self.pointDict.clear()
    self.close()

def keyPressEvent(self, e):
    if e.key() == Qt.Key_S:
        self._delegate.stop()
    elif e.key() == Qt.Key_N:
        self._delegate.changePointN()
    elif e.key() == Qt.Key_M:
        self._delegate.take_screenshots = True

def setPointObj(self, key, pointObj):
    self.lock.acquire()
    self.pointDict[key] = pointObj
    self.lock.release()

def draw(self, point, key):
    self.lock.acquire()
    if len(self.pointDict) != 0:
        self.pointDict[key].setPoint(point)
        self.update()
    self.lock.release()

def paintEvent(self, e):
    self.qp.begin(self)
    self.drawWidget(self.qp)
    self.qp.end()

def drawWidget(self, qp):
    self.lock.acquire()

    if len(self.pointDict) != 0:
        if len(self.pointDict) == 2 and self.pointDict[0].point[0] != -1 and
self.pointDict[1].point[0] != -1:
            self.error = (int(math.fabs(self.pointDict[0].point[0] -
self.pointDict[1].point[0])),
                        int(math.fabs(self.pointDict[0].point[1] -
self.pointDict[1].point[1])))
            self.countError += 1
            self.sumError = (self.sumError[0] + self.error[0], self.sumError[1] +
self.error[1])
            self.averError = (int(self.sumError[0] / self.countError),
int(self.sumError[1] / self.countError))

        for key in self.pointDict:

```



```

        self.pointDict[key].draw(qp)

    if self.startRecordingFlag:
        self.screenRecorder.writeVideo(self.pointDict, self.error, self.averError)

    self.lock.release()

```

2.4.4 Текст програми у файлі data_analyzer.py

```

from pathlib import Path
from src.calibraion.data_analyzer.experiment_data import ExperimentData
from src.calibraion.data_analyzer.data_operations import DataOperations
from src.calibraion.calibration_settings import CalibrationSettings
from src.constants.constants import Const
from src.utils.json_file import JsonFile

class DataAnalyzer:

    def __init__(self, dirPath):
        self._filePath = dirPath + '/' + Const.kAnalyzeCalibrationDataJsonFileName
        self.analyzeDataArr = self._loadAnalyzeDataFromJsonFile()

    def _loadAnalyzeDataFromJsonFile(self):
        analyzeCalibrationDataFile = Path(self._filePath)
        if analyzeCalibrationDataFile.is_file() is False:
            return []

        print("[DataAnalyzer]: Loading calibration data from file...")

        analyzeDataArr = []
        json = JsonFile.readFromFile(self._filePath)
        json = json[Const.kData]
        for cameraDataJson in json:
            expData = ExperimentData()
            paramsX = (cameraDataJson[Const.kCoordX][Const.kParams1],
                       cameraDataJson[Const.kCoordX][Const.kParams2])
            paramsY = (cameraDataJson[Const.kCoordY][Const.kParams1],
                       cameraDataJson[Const.kCoordY][Const.kParams2])
            expData.setPolinomialParameters(paramsX, paramsY)
            analyzeDataArr.append(expData)
        return analyzeDataArr

    def processCalibrationDataFromCSVFiles(self, csvFiles):
        for csvFile in csvFiles:
            expData = ExperimentData()
            expData.analyze(csvFile[0], csvFile[1])
            self.analyzeDataArr.append(expData)

        self._saveAnalyzeDataToJsonFile()
        print("[DataAnalyzer]: Finished to analyze calibration data")

    def _saveAnalyzeDataToJsonFile(self):
        json = []

```

01116130.01192-01 12 01

27

```
for i in range(len(self.analyzeDataArr)):
    cameraDataJson = {
        Const.kCoordX: {
            Const.kParams1: self.analyzeDataArr[i].polynomialParametersX[0],
            Const.kParams2: self.analyzeDataArr[i].polynomialParametersX[1]
        },
        Const.kCoordY: {
            Const.kParams1: self.analyzeDataArr[i].polynomialParametersY[0],
            Const.kParams2: self.analyzeDataArr[i].polynomialParametersY[1]
        }
    }
    json.append(cameraDataJson)

    dataJson = {
        Const.kData: json
    }

    print(dataJson)

JsonFile.writeToFile(self._filePath, dataJson)

def analyzeData(self, pairDataArr):
    monitorCoordinates = []
    for key in pairDataArr:
        monitorCoordinates.append(self.analyzeDataArr[key].processData(pairDataArr[key]))

    return self._averageMonitorCoord(monitorCoordinates)

def _averageMonitorCoord(self, monitorCoordinates):
    averMonitorCoordinates = monitorCoordinates[0]

    for i in range(len(monitorCoordinates) - 1):
        averMonitorCoordinates = DataOperations.findAver(averMonitorCoordinates,
monitorCoordinates[i + 1])

    averMonitorCoordinates =
DataOperations.stabilizeMonitorCoordinate(averMonitorCoordinates)
    return averMonitorCoordinates
```

2.4.5 Текст програми у файлі experiment_data.py

```
import numpy as np
from src.utils.csv_file import CSVFile
from src.calibraion.data_analyzer.data_operations import DataOperations

class ExperimentData:

    def __init__(self):
        self.polynomialParametersX = []
        self.polynomialParametersY = []

    def setPolinomialParameters(self, paramsX, paramsY):
        self.polynomialParametersX = paramsX
```

```

self.polynomialParametersY = paramsY

def analyze(self, filePath, delim):
    data = self._read_data(filePath, delim)
    self._findMonitorCoordinates(data)

def _read_data(self, filePath, delim):
    print("[filePath]", filePath)
    fileData = CSVFile.read_data(filePath, delim)

    # delete the first row - it's a title
    del fileData[0]

    fileData = np.array(fileData)
    return fileData.astype(np.float)

def _findMonitorCoordinates(self, data):
    controlPointsArrX, \
    leftPointsBordersArrX, \
    leftPupilCenterArrX, \
    rightPointsBordersArrX, \
    rightPupilCenterArrX = self._selectDataX(data)

    self.polynomialParametersX, \
    monitorCoordinatesX = self._doCalculation(controlPointsArrX,
                                              leftPointsBordersArrX,
                                              leftPupilCenterArrX,
                                              rightPointsBordersArrX,
                                              rightPupilCenterArrX,
                                              isCoordX=True)

    controlPointsArrY, \
    leftPointsBordersArrY, \
    leftPupilCenterArrY, \
    rightPointsBordersArrY, \
    rightPupilCenterArrY = self._selectDataY(data)

    self.polynomialParametersY, \
    monitorCoordinatesY = self._doCalculation(controlPointsArrY,
                                              leftPointsBordersArrY,
                                              leftPupilCenterArrY,
                                              rightPointsBordersArrY,
                                              rightPupilCenterArrY,
                                              isCoordX=False)

    monitorCoordinates = np.vstack((monitorCoordinatesX, monitorCoordinatesY)).T

def _selectDataX(self, data):
    data = np.array(sorted(data, key=lambda a_entry: a_entry[0]))

    # get coordinates of follow points
    controlPointsArr = data[:, 0]

```

```

# get coordinates of left eye point borders
leftPointsBordersArr = data[:, [2, 4, 6, 8, 10, 12]]

# get coordinates of left eye pupil center
leftPupilCenterArr = data[:, 14]

# skip values of pupil radius
# skip head coordinates

# get coordinates of right eye point borders
rightPointsBordersArr = data[:, [25, 27, 29, 31, 33, 35]]

# get coordinates of right eye pupil center
rightPupilCenterArr = data[:, 37]

# skip last coordinates
return controlPointsArr, leftPointsBordersArr, leftPupilCenterArr,
rightPointsBordersArr, rightPupilCenterArr

def _selectDataY(self, data):
    data = np.array(sorted(data, key=lambda a_entry: a_entry[0]))

    # get coordinates of follow points
    controlPointsArr = data[:, 1]

    # get coordinates of left eye point borders
    leftPointsBordersArr = data[:, [3, 5, 7, 9, 11, 13]]

    # get coordinates of left eye pupil center
    leftPupilCenterArr = data[:, 15]

    # skip values of pupil radius
    # skip head coordinates

    # get coordinates of right eye point borders
    rightPointsBordersArr = data[:, [26, 28, 30, 32, 34, 36]]

    # get coordinates of left eye pupil center
    rightPupilCenterArr = data[:, 38]

    # skip last coordinates
    return controlPointsArr, leftPointsBordersArr, leftPupilCenterArr,
    rightPointsBordersArr, rightPupilCenterArr

def _doCalculation(self,
                    controlPointsArr,
                    leftPointsBordersArr,
                    leftPupilCenterArr,
                    rightPointsBordersArr,
                    rightPupilCenterArr,
                    isCoordX):
    leftCenterOfEyeArr = DataOperations.findCoEArr(leftPointsBordersArr)
    rightCenterOfEyeArr = DataOperations.findCoEArr(rightPointsBordersArr)

```

```

    if isCoordX is True:
        leftPupilMovableAreaArr = DataOperations.findDiff(leftCenterOfEyeArr,
leftPupilCenterArr)
        rightPupilMovableAreaArr = DataOperations.findDiff(rightCenterOfEyeArr,
rightPupilCenterArr)
    else:
        leftPupilMovableAreaArr = DataOperations.findDiff(leftPupilCenterArr,
leftCenterOfEyeArr)
        rightPupilMovableAreaArr = DataOperations.findDiff(rightPupilCenterArr,
rightCenterOfEyeArr)

    averPupilMovableAreaArr = DataOperations.findAver(leftPupilMovableAreaArr,
rightPupilMovableAreaArr)
    averPolynomialParameters, _ =
DataOperations.findLeastSquaresPolynomial(controlPointsArr, averPupilMovableAreaArr)

    monitorPoints = DataOperations.calculateMonitorPoints(averPupilMovableAreaArr,
averPolynomialParameters)

    return averPolynomialParameters, monitorPoints

```

Process data in runtime

```

def processData(self, data):
    data = np.array([data])

    controlPointsX, \
    leftPointsBordersX, \
    leftPupilCenterX, \
    rightPointsBordersX, \
    rightPupilCenterX = self._selectDataX(data)

    controlPointsY, \
    leftPointsBordersY, \
    leftPupilCenterY, \
    rightPointsBordersY, \
    rightPupilCenterY = self._selectDataY(data)

    monitorCoordinateX = self._processDataInRuntime(leftPointsBordersX,
                                                    leftPupilCenterX,
                                                    rightPointsBordersX,
                                                    rightPupilCenterX,
                                                    self.polynomialParametersX,
                                                    isCoordX=True)

    monitorCoordinateY = self._processDataInRuntime(leftPointsBordersY,
                                                    leftPupilCenterY,
                                                    rightPointsBordersY,
                                                    rightPupilCenterY,
                                                    self.polynomialParametersY,
                                                    isCoordX=False)

    return (monitorCoordinateX, monitorCoordinateY)

```

```

def _processDataInRuntime(self,

```

01116130.01192-01 12 01

31

```
        leftPointsBorders,
        leftPupilCenter,
        rightPointsBorders,
        rightPupilCenter,
        polynomialParameters,
        isCoordX):
    leftCenterOfEye = DataOperations.findCoEArr(leftPointsBorders)
    rightCenterOfEye = DataOperations.findCoEArr(rightPointsBorders)

    if isCoordX is True:
        leftPupilMovableArea = DataOperations.findDiff(leftCenterOfEye, leftPupilCenter)
        rightPupilMovableArea = DataOperations.findDiff(rightCenterOfEye,
rightPupilCenter)
    else:
        leftPupilMovableArea = DataOperations.findDiff(leftPupilCenter, leftCenterOfEye)
        rightPupilMovableArea = DataOperations.findDiff(rightPupilCenter,
rightCenterOfEye)

    averPupilMovableArea = DataOperations.findAver(leftPupilMovableArea,
rightPupilMovableArea)
    return DataOperations.calculateMonitorPoints(averPupilMovableArea,
polynomialParameters)
```

2.5 Пакет GUI

2.5.1 Текст програми у файлі main.py

```
import time

from PyQt5.QtWidgets import QMainWindow
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QMessageBox

from ..gui.gui_utils import GuiUtils
from ..video.frame_capturer import FrameCapture
from ..gaze_tracking.head_side import Side
from ..calibraion.calibration_module import CalibrationModule
from .left_eye_settings import LeftEyeSettings
from .right_eye_settings import RightEyeSettings
from ..utils.utils import Utils
from ..constants.constants import Const
from .calibration_tab import CalibrationTab
from ..gui.settings_saver import SettingsSaver
from .information_content_tab import InformationContentTab
from ..gui.AppMode import AppMode

class Window(QMainWindow):

    def __init__(self):
        super(Window, self).__init__()
        loadUi('resources/QtGUI.ui', self)

        self.__initUI()
        self.__initObjs()
```

```
def __initUI(self):
    self.startVideoButton.clicked.connect(self.start_webcam)
    self.stopVideoButton.clicked.connect(self.stop_webcam)
    self.stopVideoButton.setEnabled(False)
    self.startTrainingButton.setEnabled(False)
    self.startDetectingPointGazeButton.setEnabled(False)

    self.showHeadPoints.hide()
    self.groupBox_l_threshold.hide()
    self.groupBox_l_flare.hide()
    self.groupBox_r_threshold.hide()
    self.groupBox_r_flare.hide()
    self.groupBox_select_next_items.hide()

def __initObjs(self):
    # Create a folder if it doesn't exist
    Utils.create_dir(Const.k_output_folder_name)

    self.frameCapture = None
    self.calibrationModule = CalibrationModule(self)

    self.left_eye_widget = LeftEyeSettings(self)
    self.right_eye_widget = RightEyeSettings(self)
    self.calibrationTab = CalibrationTab(self)
    self.pog_calibration = False
    self.currentCameraID = 0

    self.guiSettings = SettingsSaver(self)
    self.guiSettings.loadSettings()

    self.infoContentTab = InformationContentTab(self)
    self.appMode = AppMode.NoneMode

def closeEvent(self, event):
    self.guiSettings.saveSettings()
    if self.frameCapture is not None:
        self.frameCapture.stop()
        time.sleep(1)
    event.accept()

def start_webcam(self):
    try:
        self.frameCapture = FrameCapture(self.get_count_cameras(), self)
        self.frameCapture.start()
    except:
        GuiUtils.message_box("Failed to install cameras. Check your devices",
                             QMessageBox.Critical)
        return

    self.startVideoButton.setEnabled(False)
    self.stopVideoButton.setEnabled(True)
```

```
self.startTrainingButton.setEnabled(True)
self.startDetectingPointGazeButton.setEnabled(True)

def stop_webcam(self):
    self.startVideoButton.setEnabled(True)
    self.stopVideoButton.setEnabled(False)
    self.startTrainingButton.setEnabled(False)
    self.startDetectingPointGazeButton.setEnabled(False)
    self.frameCapture.stop()

def get_count_cameras(self):
    if self.oneCameraRadioButton.isChecked():
        return 1
    else:
        return 2

def take_snapshots(self, path, index):
    self.left_eye_widget.take_snapshot(path, index)
    self.right_eye_widget.take_snapshot(path, index)

'''Delegate methods:'''

def show_video_frame(self, number, frame):
    if number == 0:
        GuiUtils.show_frame(frame, self.videoFrameLabel_0)
    else:
        GuiUtils.show_frame(frame, self.videoFrameLabel_1)

def frame_size(self):
    return GuiUtils.get_label_size(self.videoFrameLabel_0)

def params_settings(self, side):
    if side == Side.Left:
        return self.left_eye_widget.params_settings
    else:
        return self.right_eye_widget.params_settings

def show_eye_frames(self, side, frames):
    if self.currentCameraID == 0:
        if side == Side.Left:
            self.left_eye_widget.show_frames(frames)
        else:
            self.right_eye_widget.show_frames(frames)

@property
def does_show_eye_contour(self):
    return self.showEyeContourBox.isChecked()
```



```

@property
def does_show_eye_pupil(self):
    return self.showEyePupilBox.isChecked()

@property
def does_show_head_area(self):
    return self.showHeadPoints.isChecked()

def display_data(self, data, side):
    if side == Side.Left:
        self.left_eye_widget.show_data(data)
    else:
        self.right_eye_widget.show_data(data)

def getExpDirName(self):
    return self.calibrationTab.getExpDirPath()

def getCSVFilesFromDir(self, dir):
    return self.calibrationTab.getCSVFilesFromDir(dir)

def processData(self, data, cameraID):
    if self.appMode == AppMode.CalibrationMode:
        self.calibrationTab.processData(data, cameraID)
    elif self.appMode == AppMode.InformationContentMode:
        self.infoContentTab.processData(data, cameraID)

```

2.5.2 Текст програми у файлі calibration_tab.py

```

from shutil import copy2
import os.path
from PyQt5.QtWidgets import QMessageBox
from ..calibraion.calibration_settings import CalibrationSettings
from ..calibraion.calibration_module import CalibrationModule

from src.utils.csv_file import CSVFile
import src.calibraion.json_generator as js
from ..utils.utils import Utils
from ..constants.constants import Const
from ..gui.gui_utils import GuiUtils
from src.point_gaze.point_gaze_detector import PointGazeDetector
from ..calibraion.data_analyzer.data_analyzer import DataAnalyzer
from .AppMode import AppMode

class CalibrationTab:
    def __init__(self, delegate):
        self.calibrationModule = delegate.calibrationModule
        self._delegate = delegate
        self.csv_files = []
        self._dirPath = None

```

```

self.pairDataDict = dict()

self.__initUI()

def __initUI(self):
    self._delegate.generateJsonButton.clicked.connect(self.generate_json)
    self._delegate.startTrainingButton.clicked.connect(self.startCalibrationProcess)

self._delegate.generateJsonMovingPointButton.clicked.connect(self.generate_moving_point_json)

self._delegate.startDetectingPointGazeButton.clicked.connect(self.startDetectingGazePoint)

def generate_json(self):
    print('[CalibrationWidget]: Start generate json')
    js.generate_json(self.point_form,
                     self._delegate.sizePointBox.value(),
                     self.point_color,
                     self._delegate.timerSpinBox.value(),
                     self._delegate.w_pointNumbersBox.value(),
                     self._delegate.h_pointNumbersBox.value(),
                     self._delegate.descriptionTextEdit.toPlainText())
    print('[CalibrationWidget]: Finish generate json')
    GuiUtils.message_box("Json file was created", QMessageBox.Information)

def generate_moving_point_json(self):
    print('[CalibrationWidget]: Start generate moving point json')
    js.generate_moving_point_json(self.point_form,
                                   self._delegate.sizePointBox.value(),
                                   self.point_color,
                                   self._delegate.timerSpinBox.value(),
                                   self._delegate.stepSpinBox.value(),
                                   self.direction,
                                   self._delegate.descriptionTextEdit.toPlainText())
    print('[CalibrationWidget]: Finish generate moving point json')
    GuiUtils.message_box("Json file was created", QMessageBox.Information)

@property
def point_form(self):
    if self._delegate.circleRadioButton.isChecked():
        return js.PointForm.Circle
    else:
        return js.PointForm.Rectangle

@property
def point_color(self):
    return (self._delegate.r_colorBox.value(),
            self._delegate.g_colorBox.value(),
            self._delegate.b_colorBox.value())

@property
def direction(self):
    if self._delegate.verticalButton.isChecked():
        return js.Direct.Vertical
    elif self._delegate.horizontalButton.isChecked():

```

```

        return js.Direct.Horizontal
    else:
        return js.Direct.Diagonal

def startCalibrationProcess(self):
    dir_name = self._delegate.expNameEdit.text()
    dirPath = self.getExpDirPath(dir_name)
    self.startCalibrationProcessWithDirPath(dirPath)

def startCalibrationProcessWithDirPath(self, dirPath):
    self._dirPath = dirPath
    self._delegate.appMode = AppMode.CalibrationMode
    self._create_csv_files(dirPath)

    self.calibrationModule.isSavedDataToFile = True
    self.calibrationModule.start(Const.k_calibration_settings_json_file_path,
                                  dirPath,
                                  self._getCSVFilePaths())

def _getCSVFilePaths(self):
    files = []
    for csvFileObj in self.csv_files:
        files.append([csvFileObj.file_name, csvFileObj.delim])
    return files

def getExpDirPath(self, dir_name):
    if len(dir_name) == 0:
        QtGuiUtils.message_box("Name of experiment is empty. Enter the data",
                                QMessageBox.Critical)
    return

    dirPath = Const.k_output_folder_name + '/' + dir_name
    Utils.create_dir(dirPath)
    Utils.create_dir(dirPath + '/' + Const.k_frames_folder_name)
    copy2(Const.k_calibration_settings_json_file_path, dirPath)
    return dirPath

def _create_csv_files(self, dir):
    self.csv_files = []
    for i in range(self._delegate.get_count_cameras()):
        filePath = dir + "/calibration_data_cameraID_" + str(i) + ".csv"
        self.csv_files.append(CSVFile(filePath, Const.k_csv_delim,
                                        Const.k_csv_title_row))

def startDetectingGazePoint(self):
    self._delegate.appMode = AppMode.CalibrationMode
    print('[CalibrationWidget]: Start detecting gaze point')
    dirPath = os.getcwd() + '/' + Const.k_output_folder_name + '/' +
    self._delegate.expNameEdit.text()
    dataAnalyzer = DataAnalyzer(dirPath)

```

```

if len(dataAnalyzer.analyzeDataArr) == 0:
    filePaths = self.getCSVFilesFromDir(dirPath)
    if filePaths is None:
        print("[CalibrationWidget]: Failed to get csv file paths")
        return

    dataAnalyzer.processCalibrationDataFromCSVFiles(filePaths)

self.calibrationModule.pointGazeDetector = PointGazeDetector(dataAnalyzer, dirPath)
self.calibrationModule.isDetectedPointGaze = True
self.calibrationModule.mode = CalibrationModule.Mode.PointOfGaze
self.calibrationModule.start(Const.k_observed_point_settings,
                             dirPath,
                             False)

def getCSVFilesFromDir(self, dir):
    calibrationSettings = None
    if os.path.exists(dir + '/' + Const.kJsonCalibrationSettingsFileName):
        calibrationSettings = CalibrationSettings(dir + '/' +
Const.kJsonCalibrationSettingsFileName)

    if calibrationSettings is None:
        return None

    files = []
    if os.path.exists(dir + '/calibration_data_cameraID_0.csv'):
        files.append((dir + '/calibration_data_cameraID_0.csv',
calibrationSettings.delim))

    if os.path.exists(dir + '/calibration_data_cameraID_1.csv'):
        files.append((dir + '/calibration_data_cameraID_1.csv',
calibrationSettings.delim))

    return files

def processData(self, data, cameraID):
    if self.calibrationModule.isDetectedPointGaze is True:
        self.pairDataDict[cameraID] = data
        if len(self.pairDataDict) == self._delegate.get_count_cameras():
            self.calibrationModule.detectPointGaze(self.pairDataDict)
            self.pairDataDict.clear()

    if self.calibrationModule.isSavedDataToFile is True and data[0] != -1:
        self.calibrationModule.save_data_to_file(data, self.csv_files[cameraID])

```

2.5.3 Текст програми у файлі information_content_tab.py

```

import os.path
import shutil
from PyQt5.QtWidgets import QFileDialog
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import Qt

```

```

from shutil import copy2
from ..utils.utils import Utils
from ..constants.constants import Const

from .information_content_widget import InformationContentWidget
from ..point_gaze.point_gaze_handler import PointGazeHandler
from ..gui.AppMode import AppMode

class InformationContentTab:
    def __init__(self, delegate):
        self._delegate = delegate
        self._delegate.chooseInfoContentFileButton.clicked.connect(self.choiceFile)
        self._delegate.resetButton.clicked.connect(self.reset)
        self._delegate.showInfoContentButton.clicked.connect(self.showInfoContent)

        self._delegate.startCalibrationButton.clicked.connect(self.startCalibration)
        self._delegate.resetCalibrationButton.clicked.connect(self.resetCalibration)

        self.infoContentWidget = InformationContentWidget()
        self.pointGazeHandler = None
        self.infoContentDirectory = None
        self.calibrationDirPath = self.getCalibrationDirPath()

    def getCalibrationDirPath(self):
        dirPath = 'output/calibration'
        Utils.create_dir(dirPath)
        Utils.create_dir(dirPath + '/' + Const.k_frames_folder_name)
        copy2(Const.k_calibration_settings_json_file_path, dirPath)
        return dirPath

    def startCalibration(self):
        shutil.rmtree(self.calibrationDirPath)
        self.calibrationDirPath = self.getCalibrationDirPath()

self._delegate.calibrationTab.startCalibrationProcessWithDirPath(self.calibrationDirPath)

    def resetCalibration(self):
        shutil.rmtree(self.calibrationDirPath)

    def choiceFile(self):
        dialog = QFileDialog()
        dialog.setFileMode(QFileDialog.DirectoryOnly)
        dialog.setOption(QFileDialog.ShowDirsOnly, False)
        dialog.exec()
        self.infoContentDirectory = dialog.directory()

self._delegate.infoContentDirPathEdit.setText(self.infoContentDirectory.absolutePath())

        self._delegate.previewInfoContentImageLabel.setScaledContents(True)

self._delegate.previewInfoContentImageLabel.setPixmap(QPixmap(self.infoContentDirectory.abso

```

```
uteFilePath('image.png'))
```

```
def reset(self):
    self._delegate.infoContentDirPathEdit.setText("")

def showInfoContent(self):
    print('[CalibrationWidget]: Show information content')
    self._delegate.appMode = AppMode.InformationContentMode
    if self.startDetectingGazePoint() == False:
        print('[InformationContentTab]: Failed to start detecting gaze point')
        return

    self.infoContentWidget.c_show(self.infoContentDirectory)

def startDetectingGazePoint(self):
    print('[InformationContentTab]: Start detecting gaze point')
    self.pointGazeHandler = PointGazeHandler(self._delegate, self.calibrationDirPath)
    return True

def processData(self, data, cameraID):
    if self.pointGazeHandler != None:
        point = self.pointGazeHandler.processData(data, cameraID)
        self.infoContentWidget.showPointGaze(point, self.selectNextItemType())

def selectNextItemType(self):
    if self._delegate.oneNextItemsRadioButton.isChecked():
        return 0
    else:
        return 1
```

2.5.4 Текст програми у файлі information_content_widget.py

```
import sys
import os
import time

from PyQt5.QtWidgets import QWidget, QLabel, QVBoxLayout, QScrollArea
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QPainter, QColor, QPixmap, QPen
from PyQt5.QtWidgets import QApplication

from src.information_content.information_content_checker import InformationContentChecker
from src.utils.csv_file import CSVFile
from src.gui.gui_utils import GuiUtils

class InformationContentWidget(QWidget):

    kObservedInformationContentResult = 'observed_information_content_result.csv'
```

```

def __init__(self):
    super().__init__()
    self.initUI()

def initUI(self):
    self.vbox = QVBoxLayout()
    self.scroll = QScrollArea()

    self.infoContentImageLabel = QLabel('Information content')
    self.infoContentImageLabel.setScaledContents(True)
    self.pixmap = None
    self.infoChecker = None

    self.csvFile = self.createCSVFile()
    self.startTime = 0
    self.flagShowPointGaze = False

def createCSVFile(self):
    if os.path.exists(InformationContentWidget.kObservedInformationContentResult):
        os.remove(InformationContentWidget.kObservedInformationContentResult)
    return CSVFile(InformationContentWidget.kObservedInformationContentResult,
                    ',',
                    ["Time",
                     "Item name",
                     "Pos",
                     "Info",
                     "Point coordinate"])

def c_show(self, infoContentDirectory):
    self.freshPixmap = QPixmap(infoContentDirectory.absoluteFilePath('image.png'))
    self.pointPixmap = QPixmap(infoContentDirectory.absoluteFilePath('image.png'))
    self.infoContentImageLabel.setPixmap(self.pointPixmap)
    self.scroll.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)
    self.scroll.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
    self.scroll.setWidgetResizable(True)
    self.scroll.setWidget(self.infoContentImageLabel)
    self.vbox.addWidget(self.scroll)

    self.vbox.setContentsMargins(0, 0, 0, 0)
    self.setLayout(self.vbox)

    self.infoChecker =
InformationContentChecker(infoContentDirectory.absoluteFilePath('information_content.json'))

    self.startTime = time.time()
    self.flagShowPointGaze = True

    self.showFullScreen()

def c_stop(self):
    self.flagShowPointGaze = False
    self.close()

```

```

def keyPressEvent(self, e):
    if e.key() == Qt.Key_S:
        self.c_stop()

def showPointGaze(self, point, selectNextItemType):
    # if self.flagShowPointGaze is False:
    #     return

    pixmap = self.pointPixmap.copy()
    size = 20

    if point[0] != -1:
        width, height = GuiUtils.monitor_size()
        if point[0] >= width:
            point = (point[0] - size * 2, point[1])

        if point[1] >= height:
            point = (point[0], point[1] - size * 2)

        newPoint = (point[0] + self.scroll.horizontalScrollBar().value(),
                    point[1] + self.scroll.verticalScrollBar().value())
        print("[InformationContentWidget] scroll position: ",
              self.scroll.horizontalScrollBar().value(),
              self.scroll.verticalScrollBar().value())

        print("[InformationContentWidget] point - ", newPoint)

        timeSecond = (time.time() - self.startTime)
        item = self.infoChecker.check(newPoint, selectNextItemType, pixmap)
        data = [timeSecond, "", "", "", newPoint]
        if item:
            data = [timeSecond, item.name, item.pos, item.info, newPoint]
        self.csvFile.write_data(data)

        painter = QPainter(pixmap)
        painter.setBrush(QColor(255, 0, 0))
        painter.setPen(QPen(Qt.black, 1, Qt.SolidLine))
        painter.drawEllipse(newPoint[0], newPoint[1], size, size)
        painter.end()

    self.infoContentImageLabel.setPixmap(pixmap)
    self.update()

```

2.5.5 Текст програми у файлі left_eye_settings.py

```

from .eye_settigns import EyeSettings

class LeftEyeSettings(EyeSettings):

    def __init__(self, delegate):
        super(LeftEyeSettings, self).__init__(delegate)

        self.originalEyeFrame = delegate.l_originalEyeFrame

```



```

self.thresholdEyeFrame = delegate.l_thresholdEyeFrame
self.thresholdValueSpinBox = delegate.l_thresholdValueSpinBox
self.binaryRadioButton = delegate.l_binaryRadioButton
self.binaryInvRadioButton = delegate.l_binaryInvRadioButton

self.histogramEqualEyeFrame = delegate.l_histogramEqualEyeFrame

self.medianBlurEyeFrame = delegate.l_medianBlurEyeFrame
self.medianBlurValueSpinBox = delegate.l_medianBlurValueSpinBox

self.contoursEyeFrame = delegate.l_contoursEyeFrame
self.pupilEyeFrame = delegate.l_pupilEyeFrame

self.point_1_label = delegate.point_36e
self.point_2_label = delegate.point_37e
self.point_3_label = delegate.point_38e
self.point_4_label = delegate.point_39e
self.point_5_label = delegate.point_40e
self.point_6_label = delegate.point_41e
self.iris_radius_label = delegate.leftEyeIrisRadius
self.pupil_center_label = delegate.leftEyePupilCenterPoint

self.eyeflareFrame = delegate.l_thresholdEyeBlickFrame
self.thresholdEyeFlareValueSpinBox = delegate.l_thresholdValueBlickSpinBox
self.binaryFlareRadioButton = delegate.l_binaryBlickRadioButton
self.binaryInvFlareRadioButton = delegate.l_binaryInvBlickRadioButton

self.side = 'Left'

```

2.5.6 Текст програми у файлі right_eye_settings.py

```

from .eye_settigns import EyeSettings

class RightEyeSettings(EyeSettings):

    def __init__(self, delegate):
        super(RightEyeSettings, self).__init__(delegate)

        self.originalEyeFrame = delegate.r_originalEyeFrame

        self.thresholdEyeFrame = delegate.r_thresholdEyeFrame
        self.thresholdValueSpinBox = delegate.r_thresholdValueSpinBox
        self.binaryRadioButton = delegate.r_binaryRadioButton
        self.binaryInvRadioButton = delegate.r_binaryInvRadioButton

        self.histogramEqualEyeFrame = delegate.r_histogramEqualEyeFrame

        self.medianBlurEyeFrame = delegate.r_medianBlurEyeFrame
        self.medianBlurValueSpinBox = delegate.r_medianBlurValueSpinBox

        self.contoursEyeFrame = delegate.r_contoursEyeFrame
        self.pupilEyeFrame = delegate.r_pupilEyeFrame

        self.point_1_label = delegate.point_42e

```

```

self.point_2_label = delegate.point_43e
self.point_3_label = delegate.point_44e
self.point_4_label = delegate.point_45e
self.point_5_label = delegate.point_46e
self.point_6_label = delegate.point_47e
self.iris_radius_label = delegate.rightEyeIrisRadius
self.pupil_center_label = delegate.rightEyePupilCenterPoint

self.eyeflareFrame = delegate.r_thresholdEyeBlickFrame
self.thresholdEyeFlareValueSpinBox = delegate.r_thresholdValueBlickSpinBox
self.binaryFlareRadioButton = delegate.r_binaryBlickRadioButton
self.binaryInvFlareRadioButton = delegate.r_binaryInvBlickRadioButton

self.side = 'Right'

```

2.5.7 Текст програми у файлі eye_settings.py

```

import numpy as np

from .gui_utils import GuiUtils

import cv2

from ..constants.constants import Const
from ..utils.utils import Utils

class EyeSettings:
    def __init__(self, delegate):
        self._delegate = delegate

        self.originalEyeFrame = None

        self.thresholdEyeFrame = None
        self.thresholdValueSpinBox = None
        self.binaryRadioButton = None
        self.binaryInvRadioButton = None

        self.histogramEqualEyeFrame = None

        self.medianBlurEyeFrame = None
        self.medianBlurValueSpinBox = None

        self.contoursEyeFrame = None
        self.pupilEyeFrame = None

        self.point_1_label = None
        self.point_2_label = None
        self.point_3_label = None
        self.point_4_label = None
        self.point_5_label = None
        self.point_6_label = None
        self.iris_radius_label = None
        self.pupil_center_label = None

        self.eyeflareFrame = None

```

01116130.01192-01 12 01

44

```
self.thresholdEyeFlareValueSpinBox = None
self.binaryFlareRadioButton = None
self.binaryInvFlareRadioButton = None

self.side = None

def take_snapshot(self, path, index):
    if self.originalEyeFrame.pixmap():
        filename = self.create_frame_name(path, index)
        GuiUtils.save_snapshot(self.originalEyeFrame.pixmap(), filename)
    else:
        print('[EyeSettings] frame doesnt exist')

def create_frame_name(self, path, index):
    filename = path + '/' + Const.k_frames_folder_name + '/' + self.side + "_frame" +
str(index) + ".png"
    print('[frame filename]', filename)
    return filename

@property
def threshold(self):
    return self.thresholdValueSpinBox.value()

@property
def thresholding_type(self):
    if self.binaryRadioButton.isChecked():
        return cv2.THRESH_BINARY
    else:
        return cv2.THRESH_BINARY_INV

def show_eye_flare_frame_filter(self, frame):
    GuiUtils.show_frame(frame, self.eyeFlareFrame)

@property
def threshold_eye_flare(self):
    return self.thresholdEyeFlareValueSpinBox.value()

@property
def threshold_eye_flare_type(self):
    if self.binaryFlareRadioButton.isChecked():
        return cv2.THRESH_BINARY
    else:
        return cv2.THRESH_BINARY_INV

@property
def params_settings(self):
    return (int(self.threshold)
        , self.thresholding_type
        , int(self.medianBlurValueSpinBox.value()))
```

```

        , int(self.threshold_eye_flare)
        , self.threshold_eye_flare_type)

def show_frames(self, frames):
    if len(frames) == 0:
        return

    GuiUtils.show_frame(frames[0], self.originalEyeFrame)
    GuiUtils.show_frame(frames[1], self.histogramEqualEyeFrame)
    GuiUtils.show_frame(frames[2], self.medianBlurEyeFrame)
    GuiUtils.show_frame(frames[3], self.thresholdEyeFrame)
    GuiUtils.show_frame(frames[4], self.contoursEyeFrame)
    GuiUtils.show_frame(frames[5], self.pupilEyeFrame)

    if len(frames) > 6:
        self.show_eye_flare_frame_filter(frames[6])

def show_data(self, data):
    # the first 12 elements are eye landmarks coordinates
    # the next 2 elements are pupil coordinates
    # the next 1 element is iris radius
    GuiUtils.show_point_to_label((data[0], data[1]), self.point_1_label)
    GuiUtils.show_point_to_label((data[2], data[3]), self.point_2_label)
    GuiUtils.show_point_to_label((data[4], data[5]), self.point_3_label)
    GuiUtils.show_point_to_label((data[6], data[7]), self.point_4_label)
    GuiUtils.show_point_to_label((data[8], data[9]), self.point_5_label)
    GuiUtils.show_point_to_label((data[10], data[11]), self.point_6_label)
    GuiUtils.show_point_to_label((data[12], data[13]), self.pupil_center_label)
    self.iris_radius_label.setText(GuiUtils.convert_data_to_str(data[14]))

```

2.6 Пакет PointGazeDetector

2.6.1 Текст програми у файлі point_gaze_detector.py

```

from src.point_gaze.point_gaze import GazePoint, PointForm
from src.constants.constants import Const
from src.utils.csv_file import CSVFile
import os

class PointGazeDetector:
    def __init__(self, dataAnalyzer, dirPath):
        self.dataAnalyzer = dataAnalyzer
        self.monitorPointGaze = GazePoint(PointForm.Circle, 20, (0, 0, 255))
        self.monitorCoordinates = None

        filePath = dirPath + '/' + Const.k_csv_pog_file
        if os.path.isfile(filePath):
            os.remove(filePath)
        self.csvFile = CSVFile(filePath, Const.k_csv_delim, Const.k_csv_pog_title_row)

    def detectPointGaze(self, pairDataDict):
        return self.dataAnalyzer.analyzeData(pairDataDict)

```

2.6.2 Текст програми у файлі point_gaze_handler.py

```

import datetime, time

from ..calibraion.data_analyzer.data_analyzer import DataAnalyzer
from src.point_gaze.point_gaze_detector import PointGazeDetector

class PointGazeHandler:

    def __init__(self, delegate, expDirectoryPath):
        self.delegate = delegate
        self.pointGazeDetector = self.createPointGazeDetector(expDirectoryPath)
        self.pairDataDict = dict()

    def createPointGazeDetector(self, expDirectoryPath):
        dataAnalyzer = DataAnalyzer(expDirectoryPath)

        if len(dataAnalyzer.analyzeDataArr) == 0:
            filePaths = self.delegate.getCSVFilesFromDir(expDirectoryPath)
            if filePaths is None:
                print("[CalibrationWidget]: Failed to get csv file paths")
                return None

            dataAnalyzer.processCalibrationDataFromCSVFiles(filePaths)

        return PointGazeDetector(dataAnalyzer, expDirectoryPath)

    def processData(self, data, cameraID):
        if self.pointGazeDetector == None:
            return (-1, -1)

        self.pairDataDict[cameraID] = data
        if len(self.pairDataDict) == self.delegate.get_count_cameras():
            point = self.pointGazeDetector.detectPointGaze(self.pairDataDict)
            print("[PointGazeHandler]: Monitor coordinates: ", point,
datetime.datetime.today())
            self.pairDataDict.clear()
            return point
        return (-1, -1)

```



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МІНІСТЕРСТВО ІНФРАСТРУКТУРИ УКРАЇНИ
ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ЗАЛІЗНИЧНОГО ТРАНСПОРТУ
ІМЕНІ АКАДЕМІКА В. ЛАЗАРЯНА

ТЕЗИ

**XIII-ої Міжнародної науково-практичної конференції
«СУЧАСНІ ІНФОРМАЦІЙНІ І КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ
НА ТРАНСПОРТІ, В ПРОМИСЛОВІСТІ ТА ОСВІТІ»
11-12 грудня 2019**

Тезисы

**XIII-й Международной научно-практической
конференции «СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ И
КОМУНИКАЦИОННЫЕ ТЕХНОЛОГИИ НА ТРАНСПОРТЕ,
В ПРОМЫШЛЕННОСТИ И ОБРАЗОВАНИИ»
11-12 декабря 2019**

ABSTRACTS

**of the XIII-th International Conference «MODERN
INFORMATION AND COMMUNICATION TECHNOLOGIES ON
A TRANSPORT, IN INDUSTRY AND EDUCATION»
11-12, December, 2019**

Дніпро
2019

Програмний комплекс із дослідження та прогнозування параметрів силових енергетичних установок локомотивів засобами експертних систем.....	131
Скалозуб В.В., Кібець Д.В., Дніпровський національний університет залізничного транспорту ім. академіка В. Лазаряна, Україна Berik Akhmetov, Al-Farabi National University, Kazakh	
Удосконалення паралельних синхронних алгоритмів оптимального планування неоднорідних потоків у мережах	132
Скалозуб В.В., Панік Л.О., Панарін О.Д. Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна, Україна Marina Skalozub, HiQ Stockholm AB, Sweden	
Інформаційна технологія з моделювання та дослідження проєктів розвитку залізничного туризму в Україні	133
Скалозуб В.В., Шашков Р.О. Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна, Україна	
Отслеживание фокуса зрения пользователя при работе с компьютером	134
Снигур Ю.А., Разносилин В.В., Дніпровський національний університет залізничного транспорту імені В.Лазаряна, Україна	
Від ТМкарти до дорожньої карти – шляхи автоматизації вантажних перевезень ASTRUM – як оптимальний маршрут в інформаційному просторі логістики.....	135
Солтисюк О.В., ТОВ «ТМСофт» Україна Лібор Белфін, JERID, Чеська Республіка	
Аналіз механізмів та ефективності спеціалізованих мов функціонального програмування	136
Сторчак І.М., Іванов О.П., Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна	
Математическая модель секционирования сортировочных путей железнодорожных станций.....	137
Терещенко Е.А., Белорусский государственный университет транспорта, Республика Беларусь	
Два способи пошуку в базі даних інформації при розв'язанні задач із семантики	138
Тимофієва Н. К., Міжнародний науково-навчальний центр інформаційних технологій та систем НАН та МОН України, Україна	
Використання Wavelet-аналізу для визначення кордонів події звукового сигналу.....	139
Царик В.Ю., Сушков О.О. Національна металургійна академія України, Україна	
Обзор преимуществ событийно-ориентированной и сервис-ориентированной архитектуры перед традиционными методами последовательной обработки событий.....	140
Цыпкина Екатерина, Германия, Нежумира О.И., Дніпровський національний університет залізничного транспорту ім. акад. Лазаряна В.А., Україна	
Інформаційна взаємодія АС ВП УЗ-Є з системою контролю параметрів роботи тепловозів «Дельта СУ».....	141
Чердніченко М.С., Гусєва В.В., Романюк Я.М., філія «ІКТБ ІТ» АТ «Укрзалізниця», Україна	

Отслеживание фокуса зрения пользователя при работе с компьютером

Снигур Ю.А., Разносилин В.В., Днепропетровский национальный университет
железнодорожного транспорта имени В.Лазаряна, Украина

На сегодняшний день актуальным способом человеко-машинного взаимодействия является отслеживание фокуса взгляда пользователя. Эта технология может применяться в медицине, маркетинге, психологии, компьютерных играх, управлении транспортными средствами, при проведении научных исследований и т.д. Анализируя фиксацию взгляда, саккады (резкие движения глаз), изменение размера зрачков, моргания и ряд других параметров исследователи получают возможность определить эффективность и эргономичность созданного информационного ресурса или продукта. В частности, измерение задержек взгляда, может быть использовано в исследованиях по оценке читабельности (англ. readability) обучающих материалов, выводимых на различные электронные устройства.

Окулография (англ. oculography или eye-tracking) - технология, позволяющая наблюдать и записывать движения глаз: расширение зрачка, его перемещение и т.д. для того чтобы определить точку фокусировки взгляда пользователя на экране монитора.

Существуют промышленные системы захвата взгляда. Это могут быть контактные линзы со встроенными зеркалами, инфракрасные подсветки, которые отражаются глазными яблоками и регистрируются видеокамерой и т.д. Однако, все эти методы требуют сложного и дорогостоящего аппаратно-программного обеспечения.

В данной работе исследуется возможность создания системы определения фокуса взгляда пользователя с использованием минимального набора аппаратных средств (например, одной или нескольких стандартных видеокамер).

На текущем этапе работы был рассмотрен ряд библиотек, с помощью которых можно захватывать видеопоток и для каждого кадра выполнять распознавание элементов лица человека. Выбор был остановлен на библиотеке «OpenCV» так как она достаточно быстро и эффективно реализует сложные алгоритмы машинного зрения. С использованием указанной библиотеки разработано ПО, которое в режиме реального времени распознает координаты размещения глаза и зрачка на видеокадре.

Положение глаза описывается 6 ключевыми точками: правый и левый угол, две верхних и две нижних точки глаза. Ключевые точки образуют сжатый шестигранник внутри которого располагается изображение глаза. Используя эти точки и алгоритм обнаружения Blob-объектов, вычисляется положение зрачка для правого и левого глаза.

Для калибровки системы распознавания фокуса взгляда разработан следующий алгоритм. Программа отслеживает нажатие клавиш «а», «b», «с», «d». Удержание одной из этих клавиш сигнализирует о том, в какой угол монитора смотрит пользователь. Таким образом, программа в режиме реального времени распределяет видеокадры на 4 группы, каждая из которых соответствует фокусировке взгляда на известном месте монитора. При этом исключаются кадры с помехами (например, моргание глаз). Данные по каждой группе усредняются и далее сравниваются между собой для того чтобы определить зависимость между известным фокусом зрения и ключевыми точками, задающими положение глаза и зрачка на видеокадре.

Разрабатываемое в данной работе программное обеспечение является необходимым инструментом для исследования и анализа фокуса взгляда пользователя при работе с различным цифровым контентом: программой, сайтом, видеоизображением, электронным документом и т.д. С его помощью возможно в режиме реального времени отслеживать зоны монитора на которых концентрируется взгляд пользователя, а также фиксировать время задержки взгляда для каждой из зон.

Дальнейший анализ полученных данных о фокусировке взгляда пользователя может проводиться как человеком-экспертом так и программой в автоматическом режиме.

Розробка засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі ПК

Снігур Ю.О., Шинкаренко В.І., Дніпровський національний університет залізничного транспорту
імені В.Лазаряна, Україна

На сьогоднішній день, у зв'язку з поширенням епідемії коронавірусу Covid-19, закриваються навчальні заклади по всьому світу. В цей час надзвичайно зростає потреба у дистанційному навчанні. Студентам важко концентрувати увагу, коли вони засвоюють матеріал самостійно. Тому актуальним є завдання покращення концентрації уваги студента під час дистанційного навчання. Для вирішення цього завдання дослідники проводять дослідження, щоб відстежити на що студент звертає увагу, що залишається поза увагою, скільки часу він витрачає на засвоєння того чи іншого матеріалу. Для проведення таких досліджень необхідні спеціальні засоби відстеження відповідності фокусу зору зі структурою інформації на моніторі.

Окулографія (айтрекінг) – процес визначення точки, на яку спрямовується погляд чи рух ока відносно голови. Дана технологія є ефективним інструментом для дослідження зорової уваги та соціальної взаємодії в різноманітних сферах навчання.

Технологія відстеження руху очей дозволяє аналізувати, як учені обробляють отриманий матеріал та куди направлена увага студентів під час навчання. Наприклад, надмірна кількість часу, яку учень витратив на обробку певного об'єму інформації може свідчити про те, що учень не розуміє або йому важко засвоїти даний матеріал. Дану інформацію можна використовувати для налаштування навчального контенту для можливості адаптувати, налаштовувати та оптимізувати процес навчання.

Метою розробки є дослідження засобів відстеження відповідності фокусу зору зі структурою інформації на моніторі з використанням апаратних засобів, які доступні кожному для покращення засвоєння інформації студентами під час дистанційного навчання.

В даній роботі, система відстеження відповідності фокусу зору зі структурою інформації на моніторі комп'ютера дозволяє зробити висновки щодо необхідності редагування інформаційного контенту, для покращення концентрації уваги студента під час дистанційного навчання.

Метод відстеження відповідності фокусу зору зі структурою інформації на моніторі відбувається в декілька етапів: отримання та обробка відеокадрів, розпізнавання рис обличчя та очей на відеокадрі, калібрування даних, підрахунок координат фокусу зору.

Для розпізнавання рис обличчя використовується метод «Активні моделі зовнішнього вигляду», оскільки він забезпечує локалізацію антропометричних точок на зображенні обличчя, в тому числі точки області розміщення очей.

Процес калібрування дозволяє підвищити точність розпізнавання координат фокусу зору. Система записує координати зіниці очей, відповідно відображеній позиції калібрувальної точки, а далі шукає залежність між цими даними. Для цього використовується модель лінійної регресії з застосуванням методу найменших квадратів. Отримавши залежність, система використовує ці дані для підрахунку координат фокусу зору.

Система відстеження фокусу зору має деякі обмежень, яких необхідно дотримуватися, задля досягнення кращої продуктивності відстеження фокусу зору.

Для дослідження відповідності фокусу зору зі структурою інформації на моніторі, система використовує спеціальний файл, який містить дані про розміщення елементів інформаційного контенту та відстежує, на який елемент інформаційного контенту сфокусовано погляд користувача у певний період часу. можуть бути використані при дослідженні поведінки користувача під час роботи з інформаційним контентом на моніторі.

Запропонована система забезпечує більшу точність відстеження фокусу зору порівняно з іншими системами.

Мониторинг фокуса зрения пользователя для контроля и совершенствования процесса дистанционного обучения

Снигур Ю.А., Разносилин В.В., Шинкаренко В.И., Днепровский национальный университет железнодорожного транспорта имени В.Лазаряна, Украина

Дистанционное обучение играет все возрастающую роль в модернизации специального и высшего образования. Это связано как развитием технических возможностей (компьютерные общественные и частные сети, например, сеть Интернет) так и с ростом числа специальностей, которые могут быть освоены удаленно, без непосредственного участия преподавателя. К таким специальностям относятся ИТ-специалисты, финансисты, менеджеры и многие другие специальности из сферы услуг.

В данной работе предлагается способ точной идентификации действий пользователя (учащегося) при работе с обучающими материалами учебного курса. В качестве дополнительной опции предусматривается возможность анализа процесса взаимодействия учащегося с обучающим материалом. При этом могут быть рассмотрены индивидуальные и групповые паттерны взаимодействия, что в будущем позволит выполнять объективно обоснованную коррекцию обучающих материалов с целью повышения их *показателя читабельности*.

Для достижения указанной выше цели была использована технология отслеживания фокуса взгляда пользователя. Анализируя фиксацию взгляда, саккады (резкие движения глаз), изменение размера зрачков, моргания и ряд других параметров возможно определить эффективность и эргономичность созданного обучающего ресурса или продукта.

Окулография (англ. oculography или eye-tracking) - технология, позволяющая наблюдать и записывать движения глаз: расширение зрачка, его перемещение и т.д. для того чтобы определить точку фокусировки взгляда пользователя на экране монитора [1].

Существуют промышленные системы захвата взгляда. Это могут быть контактные линзы со встроенными зеркалами, инфракрасные подсветки, которые отражаются глазными яблоками и регистрируются видеокамерой и т.д. Однако, все эти методы требуют сложного и дорогостоящего аппаратно-программного обеспечения и неприменимы в условиях дистанционного обучения. Исходя из этого была исследована возможность создания системы определения фокуса взгляда пользователя с использованием минимального набора аппаратных средств (например, стандартной видеокамеры современного ноутбука).

В качестве усредненных условий дистанционного обучения была рассмотрена следующая ситуация. Пользователь читает обучающие материалы с экрана ноутбука с разрешением 1366x768 пикселей, с физическими размерами 340x200 мм. Видеокамера расположена в верхней части экрана по вертикали и по центру по горизонтали. Расстояние от экрана до переносицы пользователя равно 500 мм, он сидит неподвижно, по центру относительно экрана ноутбука (по горизонтали). Среднее расстояние между зрачками принимали равным 64 мм. Значительные (более 10 градусов) повороты и наклоны головы в данной работе не рассматривались. В описанных выше условиях контур лица пользователя занимает примерно 50% кадра по высоте и около 25-30% кадра по ширине, что делает возможным достаточно точное распознавание как контура лица в целом так и отдельных его частей (контуры глаз, зрачки, виски, переносица и т.д.).

Для технической реализации программного обеспечения по распознаванию фокуса взгляда был рассмотрен ряд библиотек, с помощью которых можно захватывать видеопоток и для каждого кадра выполнять распознавание элементов лица человека. Выбор был остановлен на библиотеке «OpenCV» так как она достаточно быстро и эффективно реализует сложные алгоритмы машинного зрения [2].

Положение каждого глаза описывается 6 ключевыми точками: правый и левый угол, две верхних и две нижних точки глаза. Ключевые точки образуют сжатый шестигранник внутри которого располагается изображение глаза (см. рис 1). Используя эти точки и алгоритм обнаружения Blob-объектов, вычисляется положение зрачка для правого и левого глаза.

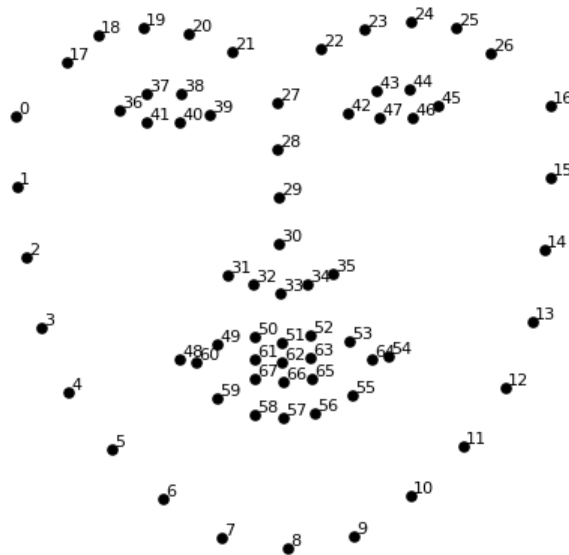


Рисунок 1. Ключевые точки человеческого лица распознаваемые библиотекой «OpenCV»

Зрачок – это самая темная центральная часть глаза. Для того чтобы определить его положение на видеокadre вырезается область размещения глаза с общего кадра. Для этого используются координаты точек 36..41 и 42..47 (см. рис. 1). Затем полученное изображение преобразуется к черно-белому с заданным порогом. На данном этапе работы пороговое значение выбиралось вручную в зависимости от условий освещения.

Далее выполняется ряд морфологических преобразований. Это простые операции, основанные на форме изображения. Для использования таких операций, требуется бинарное изображение и структурный элемент (ядро).

Ядро определяет характер операции. Ядро скользит по изображению, пиксель в исходном изображении («1» или «0») будет считаться «1», только если все пиксели под ядром равны «1», в противном случае он равен «0». В данной работе использовалось ядро в виде эллипса.

Использовались следующие морфологические преобразования:

Размытие (Erosion) – размытие контуров объектов переднего плана. Все пиксели вблизи границы будут отбрасываться в зависимости от ядра. Таким образом, уменьшается белая область в изображении или уменьшается размер объекта переднего плана.

Расширение (Dilation) – используется при удалении шумов, после размытия. Принцип работы следующий: пиксель равен «1», если хотя бы один пиксель под ядром равен «1». Таким образом, увеличивается белая область в изображении или увеличивается размер объекта переднего плана.

Следующим этапом, является поиск контуров в изображении. Контур – это кривая, соединяющая все непрерывные точки вдоль границы, имеющие одинаковые свойства (цвет, интенсивность). После нахождения всех контуров, необходимо их отсортировать, так как контур зрачка является максимальным в текущем кадре. Далее нужно найти координаты центральной точки и радиус найденного контура зрачка. Для этого необходимо использовать оптимизированный по скорости алгоритм Дугласа – Пекера [3], с помощью которого можно преобразовать найденный контур к форме зрачка.

При обнаружении зрачка для каждого захваченного кадра могут возникнуть проблемы, связанные с точностью распознавания. Чтобы решить данную проблему, необходимо стабилизировать процесс: для нескольких кадров найти усредненные координаты зрачка и считать их окончательными координатами зрачка в конце серии кадров. Число кадров, по которым производится усреднение, было определено экспериментальным путем и равно 4.

Для калибровки системы распознавания фокуса взгляда разработан следующий алгоритм. Программа загружает калибровочные данные из файла формата «json» и начинает демонстрировать в заданных точках экрана монитора курсор в виде фигуры (круг или квадрат) указанного цвета. Перемещение курсора в следующую точку происходит либо автоматически по истечению указанного интервала времени или по нажатию клавиши пользователем. В процессе калибровки пользователь должен внимательно следить за перемещением курсора.

После окончания процедуры калибровки программа записывает на диск файл в формате «csv» в котором сохранено положение курсора в каждом видеокадре и связанные с ним координаты ключевых точек глаза (см. выше).

Ниже описан алгоритм обработки результатов калибровочного теста для построения зависимости между положением зрачка на видеокадре и зоной фокуса зрения на экране монитора. Этот алгоритм выполняется отдельно для каждой из координат (по вертикали и по горизонтали).

Вначале для левого и правого зрачка вычисляем отклонение его положения от положения переносицы (см. рис. 1, точка 27). Далее определяем минимум отклонения и вычитается из полученного ранее набора значений. Таким образом, мы получаем отклонение координат зрачков от переносицы, смещенное к нулевой точке. Далее используя эти числа в качестве значений Y , а известные координаты фокуса зрения в качестве значений X , аппроксимируем набор данных линейной функцией, а также вычисляем коэффициент корреляции. Сравниваем полученные коэффициенты корреляции и выбираем в качестве базовой зависимости данные зрачка, для которого коэффициент корреляции выше.

На рис 2, в качестве примера, показаны результаты расчета для случая перемещения фокуса зрения вдоль горизонтальной оси монитора. При этом координата вдоль вертикальной оси фиксирована по центру экрана. Для обоих зрачков получены достаточно устойчивые зависимости между отклонением положения зрачка от переносицы. В данном случае в качестве опорной будет выбрана функция, соответствующая левому зрачку, т.к. для него коэффициент корреляции выше чем для правого зрачка.

Используя полученную функцию, мы можем применить обратное преобразование отклонения зрачка от переносицы в экранные координаты. На рис. 3 показано распределение погрешности такого обратного преобразования. Из представленной гистограммы видно, что с вероятностью около 70% ошибка не превышает 100 пикселей, что делает возможным идентифицировать фокус взгляда пользователя с точностью до 8-10 зон разбиения экрана ноутбука. При использованном разрешении равном 1366 пикселей ширина каждой такой зоны составляет около 136 пикселей, что вполне достаточно для идентификации контента на котором сосредоточено внимание пользователя в данный момент времени.

Несколько хуже ситуации с распознаванием точки (зоны) фокуса по вертикальной оси. Это объясняется тем, что линия глаз пользователя (линия переносицы) как правило выше видеокамеры, т.е. смещена относительно центра экрана. Физическая высота монитора примерно в два раза меньше чем его ширина. Это приводит к тому, что угол наклона лица и угол поворота глазных яблок при перемещении фокуса от верхней крайней точки монитора до нижней крайней точки монитора существенно меньше чем в случае с горизонтальной осью. Однако даже в таком случае возможно распознавание фокуса взгляда

пользователя с точностью до экрана разделенного на три зоны высотой по 265 пикселей, что на данном этапе работы еще не дает уверенной идентификации контента с которым работает пользователь.

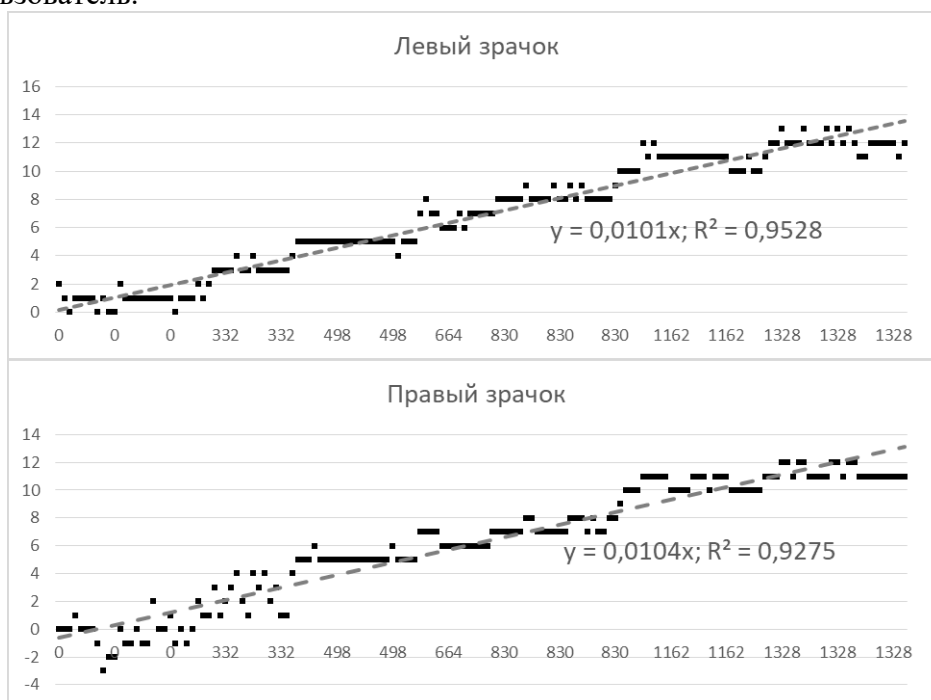


Рисунок 2. Зависимость положения зрачков и точки фокуса зрения (по горизонтальной оси)

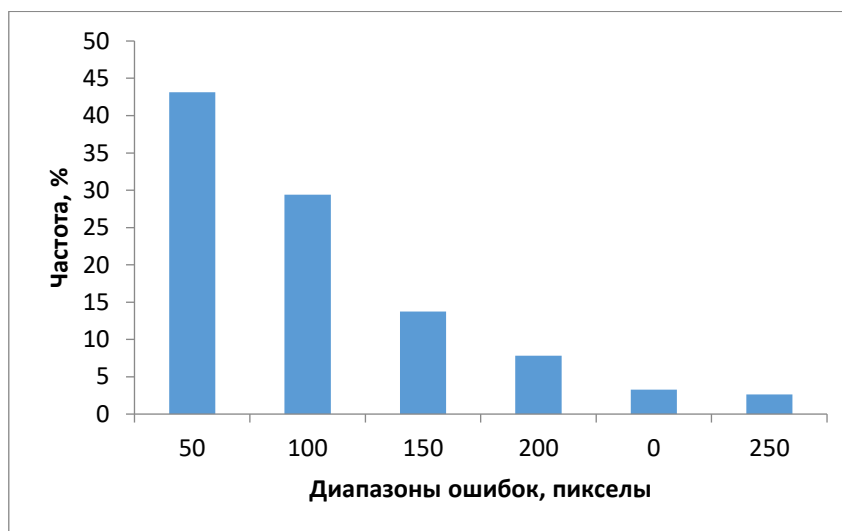


Рисунок 3. Распределение погрешности оценки координат точки фокуса зрения (по горизонтальной оси)

Таким образом, на данный момент продолжается разработка программного обеспечения, которое в домашних условиях (а это требование для дистанционного обучения) позволит идентифицировать фокус взгляда пользователя с некоторой погрешностью, фиксировать время фокусировки на той или иной области экрана и, в конечном итоге, выполнять привязку фокуса взгляда пользователя к тому контенту (абзац, рисунок, формула) с которым он работает.

В дальнейшем предполагается усовершенствовать алгоритм идентификации координат зрачка и *блика* на глазном яблоке с целью улучшить точность распознавания фокуса взгляда по вертикальной оси, а также в различных условиях освещения.

Список литературы:

1. Г.И. Фазылзянова, В.В. Балалов Применение метода айтрекинга для оценки качества графической и мультимедийной продукции // Science and world. – 2014. №3 (7), Vol. III. – С. 172-178.
2. Полякова А.С. О применении библиотеки OpenCV в задаче распознавания лиц по их изображению // Актуальные проблемы авиации и космонавтики. – 2016. №12. – С. 558-559.
3. J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line simplification algorithm. Technical report, Vancouver, BC, Canada, Canada, – 1992.