ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмний застосунок розпізнавання зображень на основі методів комп'ютерного зору»

зі спеціальності *126 Інформаційні системи та технології*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційні системи та технології*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____Віолетта ФОМЕНКО
(підпис)

Виконала: здобувачка вищої освіти групи ІСД-42
Віолетта ФОМЕНКО
(прізвище, ім'я)

Керівник PhD доцент, Валентина ДАНИЛЬЧЕНКО
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти Бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
Каміла Сторчак
“___” _____ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Віолетта Володимирівна Фоменко
(прізвище, ім’я)

1. Тема кваліфікаційної роботи: «Програмний застосунок розпізнавання зображень на основі методів комп’ютерного зору»

керівник кваліфікаційної роботи PhD доцент, Валентина Данильченко
(прізвище, ім’я, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «___» _____ 2026 року № ____.

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 01.06.2026 р.

3. Вихідні дані до кваліфікаційної роботи
– інформаційні ресурси та програмні компоненти системи розпізнавання зображень;
– модулі обробки графічних даних у середовищі програмування Python, механізми роботи з бібліотекою OpenCV для аналізу зображень і визначення контурів об’єктів;
– нормативні та методичні матеріали щодо використання сучасних технологій комп’ютерного зору, методів обробки та аналізу цифрових зображень, а також підходів до створення програмних систем розпізнавання об’єктів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
– Аналіз предметної області та постановка задачі створення програмного застосунку розпізнавання зображень на основі методів комп’ютерного зору.

– Огляд існуючих алгоритмів і технологій обробки зображень та аналіз можливостей сучасних програмних бібліотек для реалізації систем комп’ютерного зору.

– Розробка, програмна реалізація, тестування та оцінювання ефективності роботи програмного застосунку розпізнавання зображень.

5. Перелік графічного матеріалу

6. Дата видачі завдання 20.02.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз предметної області та огляд сучасних технологій комп’ютерного зору і систем обробки зображень	15.04.2026 р.	виконано
2	Вивчення технічної документації бібліотек OpenCV та PyQt і середовища програмування Python для розробки систем комп’ютерного зору	20.04.2026 р.	виконано
3	Розробка архітектури програмної системи: модулі обробки зображень, аналізу контурів та взаємодії з графічним інтерфейсом	05.05.2026 р.	виконано
4	Програмна реалізація застосунку мовою Python із використанням бібліотеки OpenCV для розпізнавання геометричних об’єктів	15.05.2026 р.	виконано
5	Реалізація та налаштування алгоритмів обробки зображень, тестування системи на різних наборах графічних даних	25.05.2026 р.	виконано
6	Завершальне тестування програмного застосунку, перевірка функціональності та підготовка демонстраційних матеріалів	30.05.2026 р.	виконано

Здобувач вищої освіти _____ Віолетта ФОМЕНКО
(підпис) *(ім’я, прізвище)*

Керівник кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(ім’я, прізвище)*

РЕФЕРАТ

Текстова частина бакалаврської роботи: 59 сторінок, 26 рисунків, 3 таблиці 32 джерел.

Об'єкт дослідження – процеси автоматизованої обробки та аналізу графічних даних із застосуванням технологій комп'ютерного зору.

Предмет дослідження – алгоритмічні методи та програмні інструменти, що забезпечують розпізнавання геометричних об'єктів на зображеннях у режимі реального часу з використанням бібліотеки OpenCV та графічного інтерфейсу, створеного на основі PyQt.

Мета роботи – створення програмної системи для автоматичного розпізнавання зображень, яка здійснює обробку відеопотоку з вебкамери або згенерованих сцен, визначає типи геометричних фігур та відображає інформацію про них у зручному інтерфейсі користувача.

Методи дослідження:

1. Дослідження сучасних підходів до реалізації систем комп'ютерного зору із використанням мови програмування Python та бібліотеки OpenCV.
2. Розроблення алгоритму виділення контурів та класифікації геометричних об'єктів на основі аналізу кількості вершин і характеристик форми.
3. Реалізація програмного інтерфейсу користувача для генерації тестових сцен та управління процесом обробки зображень.
4. Проведення експериментального тестування системи на відеопотоці з вебкамери та штучно створених зображеннях із подальшим аналізом результатів розпізнавання.

КОМП'ЮТЕРНИЙ ЗІР, АНАЛІЗ ЗОБРАЖЕНЬ, OPENCV, PYTHON, ВЕБКАМЕРА, PYQT, ВИЯВЛЕННЯ КОНТУРІВ, РОЗПІЗНАВАННЯ ФІГУР, ВІЗУАЛІЗАЦІЯ ДАНИХ, ГРАФІЧНИЙ ІНТЕРФЕЙС

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасному інформаційному суспільстві значна частина даних має візуальний характер, тому ефективна обробка та аналіз зображень набувають особливої важливості. Технології комп'ютерного зору активно застосовуються у різних сферах діяльності, зокрема у системах відеоспостереження, робототехніці, медичній діагностиці, промисловій автоматизації, безпілотних транспортних засобах та системах безпеки. Використання таких технологій дозволяє автоматизувати процеси аналізу візуальної інформації та підвищити швидкість і точність прийняття рішень.

Розпізнавання об'єктів на зображеннях у режимі реального часу – передбачає застосування алгоритмів обробки зображень, які дозволяють виділяти контури об'єктів, аналізувати їх форму та класифікувати їх відповідно до визначених характеристик. Реалізація таких алгоритмів у програмних системах забезпечує можливість створення інтелектуальних застосунків, здатних автоматично визначати типи об'єктів на зображеннях або відеопотоці.

Актуальність теми дослідження зумовлена необхідністю розроблення ефективних програмних засобів для автоматизованого розпізнавання об'єктів на зображеннях, що дозволяють застосовувати методи комп'ютерного зору для вирішення практичних задач аналізу візуальної інформації.*Об'єкт дослідження* – процеси автоматизованої обробки та аналізу графічних даних із застосуванням технологій комп'ютерного зору..

Предмет дослідження – алгоритмічні методи та програмні інструменти, що забезпечують розпізнавання геометричних об'єктів на зображеннях у режимі реального часу з використанням бібліотеки OpenCV та графічного інтерфейсу, створеного на основі PyQt.

Мета роботи – створення програмної системи для автоматичного розпізнавання зображень, яка здійснює обробку відеопотоку з вебкамери або згенерованих сцен, визначає типи геометричних фігур та відображає інформацію про них у зручному інтерфейсі користувача.

Наукові завдання:

1. Провести аналіз предметної області та дослідити сучасні підходи до обробки та розпізнавання зображень у системах комп’ютерного зору.
2. Проаналізувати існуючі алгоритми та методи виявлення об’єктів на цифрових зображеннях і визначити найбільш ефективні з них для реалізації у програмному застосунку.
3. Дослідити можливості використання мови програмування Python та бібліотеки OpenCV для реалізації алгоритмів обробки зображень і розпізнавання об’єктів.
4. Розробити алгоритм визначення геометричних фігур на зображеннях на основі аналізу контурів та характеристик форми об’єктів.
5. Спроєктувати архітектуру програмного застосунку для розпізнавання зображень із використанням графічного інтерфейсу користувача.
6. Реалізувати програмний застосунок із використанням бібліотек OpenCV та PyQt для обробки відеопотоку та візуалізації результатів розпізнавання.
7. Провести тестування розробленої системи та оцінити ефективність її роботи під час обробки зображень та відеопотоку.

Методи дослідження:

1. Дослідження сучасних підходів до реалізації систем комп’ютерного зору із використанням мови програмування Python та бібліотеки OpenCV.
2. Розроблення алгоритму виділення контурів та класифікації геометричних об’єктів на основі аналізу кількості вершин і характеристик форми.
3. Реалізація програмного інтерфейсу користувача для генерації тестових сцен та управління процесом обробки зображень.
4. Проведення експериментального тестування системи на відеопотоці з вебкамери та штучно створених зображеннях із подальшим аналізом результатів розпізнавання.

Наукова новизна одержаних результатів. У ході дослідження запропоновано підхід до розпізнавання геометричних об’єктів на основі поєднання методів попередньої обробки зображень, алгоритмів виділення меж (Canny) та аналізу

контурів. На відміну від існуючих підходів, реалізовано структуровану модульну архітектуру системи, що забезпечує ефективне поєднання алгоритмічної обробки зображень та інтерактивного графічного інтерфейсу користувача. Отримані результати підтверджують можливість досягнення достатньої точності розпізнавання об’єктів у режимі реального часу.

Практична значущість одержаних результатів. Розроблений програмний застосунок може бути використаний для розпізнавання геометричних фігур у відеопотоці, що має практичне значення для навчальних систем, систем відеоспостереження та автоматизованого аналізу зображень. Використання бібліотек OpenCV та PyQt дозволяє реалізувати зручний інтерфейс користувача та забезпечити ефективну обробку даних, що робить систему придатною для подальшого розвитку та інтеграції у прикладні програмні рішення.

Апробація результатів та публікації. Основні положення і результати бакалаврської роботи доповідались на науково практичних конференціях, що проходили на базі Державного університету інформаційно-комунікаційних технологій. III Всеукраїнська науково-технічна конференція, ДУІКТ, «Застосування машинного навчання у системах відеоаналітики та розпізнавання облич для забезпечення безпеки» на тему: «Застосування машинного навчання у системах відеоаналітики та розпізнавання облич для забезпечення безпеки». VII Міжнародній науково-технічній конференції «Сучасний стан та перспективи розвитку IoT» на тему: «Застосування методів комп’ютерного зору для автоматизованого аналізу візуальних даних».

1 ТЕОРЕТИЧНІ ОСНОВИ ОБРОБКИ ТА РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ У СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ

1.1 Розвиток технологій комп'ютерного зору, сфери їх застосування та значення у візуальному аналізі даних

Комп'ютерний зір є одним із напрямів сучасних інформаційних технологій та штучного інтелекту, який спрямований на створення методів і програмних засобів для автоматичного аналізу та інтерпретації візуальної інформації. Основною метою комп'ютерного зору є надання комп'ютерним системам здатності отримувати, обробляти та "розуміти" зображення і відеодані подібно до людського зору [1].

У загальному випадку комп'ютерний зір передбачає перетворення вхідних графічних даних у форму, придатну для подальшого аналізу, виділення характерних ознак об'єктів та їх подальшу ідентифікацію. До таких ознак можуть належати контури, текстури, кольорові характеристики, геометричні параметри та інші властивості зображення. На основі отриманої інформації система здатна визначати наявність об'єктів, їх форму, положення та інші характеристики (рис. 1.1).

Рис. 1.1 – Схема роботи системи комп'ютерного зору

Розвиток комп'ютерного зору відбувався поступово — від використання відносно простих алгоритмів обробки зображень до впровадження складних моделей штучного інтелекту, здатних аналізувати візуальні дані з високою точністю. На початкових етапах розвитку цієї галузі основна увага приділялася базовим методам цифрової обробки зображень. До них належали перетворення

кольорових зображень у відтінки сірого, фільтрація шумів, підвищення контрастності, виділення меж об’єктів, пошук контурів та визначення простих геометричних характеристик. Такі підходи дозволяли виконувати елементарне розпізнавання об’єктів, однак їх ефективність значною мірою залежала від якості зображення, освітлення, фону та інших зовнішніх факторів. Сфери застосування технологій комп’ютерного зору є різноманітними та охоплюють ключові напрями сучасної цифрової діяльності (рис. 1.2).

Рис. 1.2 – Сфери застосування комп’ютерного зору

Подальший розвиток комп’ютерного зору був пов’язаний із переходом до методів, що базувалися на аналізі ознак. На цьому етапі системи вже не лише обробляли зображення на рівні пікселів, а й виділяли характерні особливості об’єктів: кути, текстури, локальні ключові точки, співвідношення сторін, форму, контрастні області [2]. Це дало змогу покращити точність класифікації та зробити розпізнавання більш стійким до змін масштабу, повороту чи часткового перекриття об’єктів. Такі підходи тривалий час активно використовувалися в задачах пошуку об’єктів, аналізу сцен та автоматизованого контролю.

Справжній прорив у розвитку комп’ютерного зору відбувся із появою методів машинного навчання, а згодом — глибокого навчання. Якщо раніше розробник мав самостійно визначати, які саме ознаки є важливими для розпізнавання, то сучасні нейронні мережі можуть автоматично виявляти складні закономірності у великих наборах зображень. Завдяки цьому комп’ютерні системи навчилися значно точніше розпізнавати об’єкти, обличчя, текст, дорожні знаки, медичні аномалії та інші складні візуальні структури. Використання штучного інтелекту дало можливість перейти від обробки окремих статичних зображень до повноцінного аналізу відеопотоків у режимі реального часу, що суттєво розширило сферу застосування комп’ютерного зору.

Сьогодні технології комп’ютерного зору активно використовуються у системах відеоспостереження. У цій сфері вони забезпечують автоматичне виявлення руху, ідентифікацію об’єктів у кадрі, розпізнавання облич, номерних знаків транспортних засобів, контроль периметра та фіксацію підозрілих подій. На відміну від традиційного спостереження, де аналіз відео здійснюється людиною, сучасні інтелектуальні системи можуть безперервно обробляти великі обсяги відеоданих, оперативно виявляти відхилення та повідомляти оператора про потенційно небезпечні ситуації.

В медицині, комп’ютерний зір застосовують для аналізу рентгенівських знімків, комп’ютерної томографії, магнітно-резонансної томографії, ультразвукових зображень та мікроскопічних даних. Такі системи допомагають виявляти патологічні зміни, аналізувати структуру тканин, знаходити новоутворення та інші медичні ознаки, які можуть бути складними для швидкого візуального визначення. У цьому випадку комп’ютерний зір виступає не заміною лікаря, а інструментом підтримки прийняття рішень, що підвищує точність діагностики та дозволяє зменшити ймовірність помилки [3].

Також сферою використання комп’ютерного зору є автомобільна галузь, зокрема системи автономного керування транспортом. У безпілотних автомобілях камери разом з алгоритмами комп’ютерного зору забезпечують розпізнавання дорожньої обстановки: виявлення смуг руху, пішоходів, транспортних засобів,

дорожніх знаків, світлофорів та перешкод. На основі аналізу цих даних система може оцінювати ситуацію на дорозі та приймати рішення щодо подальшого руху. Саме тому комп'ютерний зір є однією з ключових технологій для розвитку інтелектуальних транспортних систем.

Окремо слід відзначити застосування комп'ютерного зору в галузі безпеки. Йдеться не лише про фізичну безпеку об'єктів, а й про контроль доступу, біометричну ідентифікацію, моніторинг поведінки людей у публічних або критично важливих зонах, виявлення небезпечних предметів чи порушень встановлених правил. У поєднанні з аналітичними системами комп'ютерний зір дозволяє підвищити рівень захисту підприємств, транспортної інфраструктури, банківських установ, державних організацій та інших об'єктів, де потрібен постійний контроль візуальної інформації.

Важливість комп'ютерного зору зумовлена тим, що сучасний світ генерує величезну кількість візуальних даних, які людина не може ефективно обробляти самостійно. Камери спостереження, мобільні пристрої, медичні прилади, виробничі системи та транспортні засоби щоденно створюють великі масиви зображень і відео, аналіз яких потребує автоматизованих рішень. Саме комп'ютерний зір дає змогу перетворити ці дані на корисну інформацію, прискорити прийняття рішень, знизити вплив людського фактора та підвищити ефективність роботи різних систем.

Розвиток комп'ютерного зору від простих алгоритмів обробки зображень до інтелектуальних моделей штучного інтелекту зробив цю технологію однією з найбільш перспективних у сучасній інформатиці. Її значення постійно зростає, оскільки автоматизований аналіз візуальної інформації стає необхідним елементом функціонування багатьох сучасних інформаційних систем і цифрових сервісів.

1.2 Основні алгоритми та методи визначення об'єктів на цифрових зображеннях

У системах комп'ютерного зору для розпізнавання об'єктів застосовується

сукупність алгоритмів обробки зображень, які дозволяють підготувати вхідні дані до подальшого аналізу та виділити ключові ознаки об’єктів. До базових методів належать попередня обробка зображення, виділення меж об’єктів, бінаризація та пошук контурів. Процес розпізнавання об’єктів на зображеннях передбачає послідовне застосування ряду алгоритмів обробки, кожен з яких виконує окрему функцію підготовки та аналізу даних. Узагальнену послідовність основних етапів обробки зображення у системах комп’ютерного зору наведено на рис. 1.4.

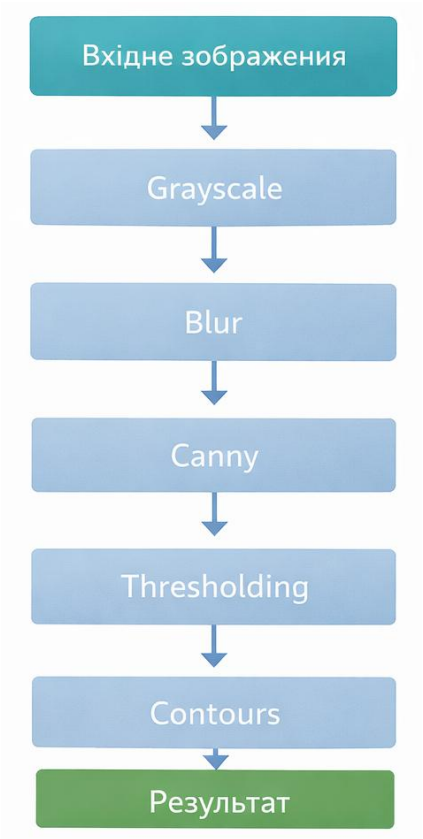


Рис. 1.4 – Загальна схема алгоритму обробки зображення

Одним із перших етапів обробки є перетворення зображення у відтінки сірого (grayscale). Це дозволяє зменшити кількість оброблюваних даних та спростити подальший аналіз, оскільки замість трьох кольорових каналів використовується один. Після цього часто застосовується згладжування зображення (blur), наприклад за допомогою гаусового фільтра. Метою цього етапу є зменшення впливу шумів та дрібних нерівностей, які можуть негативно впливати на точність виявлення об’єктів.

Наступним етапом є виділення меж об’єктів (edge detection). Одним із

поширених алгоритмів є алгоритм Кенні (Canny), який дозволяє ефективно визначати границі об’єктів на зображенні. Він базується на обчисленні градієнтів яскравості та виділенні ділянок із різкими змінами інтенсивності, що відповідають контурам об’єктів.

Для подальшого аналізу часто використовується порогова обробка (thresholding), яка перетворює зображення у бінарний вигляд. У результаті кожен піксель набуває значення 0 або 1 залежно від заданого порогу яскравості. Це дозволяє чітко відокремити об’єкти від фону та спростити процедуру їх ідентифікації.

Пошук контурів (contours). Контур представляє собою замкнену лінію, що описує межі об’єкта. Використовуючи алгоритми пошуку контурів, можна визначити форму об’єкта, його площу, периметр та інші геометричні характеристики. Саме на основі аналізу контурів здійснюється класифікація об’єктів, наприклад визначення геометричних фігур за кількістю вершин або співвідношенням сторін.

Після виділення контурів об’єктів наступним етапом є їх класифікація, яка дозволяє визначити тип об’єкта на основі його геометричних характеристик. У найпростіших системах комп’ютерного зору класифікація здійснюється за формою об’єкта та кількістю його вершин. Такий підхід є ефективним для задач розпізнавання базових геометричних фігур [5].

Зокрема, аналізуючи контур об’єкта, можна визначити кількість його кутів (вершин), що дозволяє класифікувати фігуру. Наприклад, об’єкт із трьома вершинами ідентифікується як трикутник, із чотирма — як квадрат або прямокутник (залежно від співвідношення сторін), а об’єкти з великою кількістю точок контуру можуть розглядатися як коло або багатокутник складної форми. Додатково враховуються такі параметри, як площа, периметр, співвідношення сторін та ступінь округлості об’єкта.

У більш складних випадках, коли об’єкти мають складну форму або значні варіації у зовнішньому вигляді, застосовуються методи машинного навчання. Вони дозволяють автоматично навчати модель на основі великої кількості прикладів

зображень і забезпечують більш точне розпізнавання об’єктів у складних умовах. Зокрема, сучасні підходи використовують нейронні мережі, які здатні самостійно виділяти важливі ознаки та виконувати класифікацію без явного задання правил.

Класифікація об’єктів може здійснюватися як за допомогою простих геометричних методів, так і з використанням алгоритмів машинного навчання, що значно розширює можливості систем комп’ютерного зору у задачах розпізнавання.

1.3 Аналіз можливостей бібліотек OpenCV та PyQt для створення програмних систем комп’ютерного зору

Бібліотеки OpenCV та PyQt є інструментами для створення програмних систем комп’ютерного зору, оскільки забезпечують поєднання функціональних можливостей обробки зображень із засобами побудови зручного графічного інтерфейсу користувача. Використання саме цих технологій є доцільним у процесі розроблення програмних застосунків, орієнтованих на аналіз візуальної інформації, оскільки вони дозволяють реалізувати як алгоритмічну, так і прикладну частину системи в межах одного програмного середовища.

OpenCV є однією з найпоширеніших бібліотек для роботи із цифровими зображеннями та відео. Вона містить значну кількість готових функцій і алгоритмів, що призначені для реалізації задач комп’ютерного зору, зокрема попередньої обробки зображень, фільтрації, бінаризації, виділення контурів, пошуку ключових точок, сегментації, аналізу форми об’єктів і розпізнавання візуальних структур. Важливою перевагою OpenCV є її орієнтація на практичне використання у прикладних системах, де необхідна висока швидкість обробки графічних даних та можливість роботи у режимі реального часу [6].

Застосування OpenCV у програмних системах комп’ютерного зору є ефективним під час розроблення рішень, пов’язаних із розпізнаванням геометричних об’єктів. Засоби бібліотеки дозволяють здійснювати перетворення кольорового зображення у відтінки сірого, виконувати згладжування, виділяти межі об’єктів за допомогою алгоритму Canny, застосовувати порогову обробку та

знаходити контури. На основі отриманих контурів можна визначати кількість вершин, площу, периметр та інші характеристики, необхідні для класифікації фігур. Таким чином, OpenCV забезпечує повний набір базових засобів, необхідних для реалізації алгоритмічної частини системи розпізнавання.

OpenCV є її сумісність із мовою програмування Python, що значно спрощує процес розроблення. Python має зрозумілий синтаксис, велику кількість додаткових бібліотек та активно використовується у сфері штучного інтелекту, аналізу даних і комп'ютерного зору. Поєднання Python та OpenCV дозволяє створювати гнучкі програмні рішення, які легко модифікувати, тестувати та розширювати відповідно до поставлених завдань.

Бібліотека PyQt використовується для створення графічного інтерфейсу користувача. Її застосування є важливим у тих випадках, коли програмна система повинна не лише виконувати внутрішню обробку даних, а й забезпечувати зручну взаємодію з користувачем. За допомогою PyQt можна створювати вікна програми, кнопки, поля введення, елементи керування, графічні області для відображення зображень та інші компоненти інтерфейсу. Це дозволяє реалізувати повноцінний програмний застосунок, у якому користувач може керувати процесом аналізу зображень, запускати відеопотік, переглядати результати розпізнавання та взаємодіяти з системою в реальному часі.

Перевагою PyQt є також можливість створення інтерфейсів із чіткою структурою та високим рівнем зручності використання. Для систем комп'ютерного зору це має особливе значення, оскільки результати аналізу повинні бути представлені користувачу наочно, зрозуміло та оперативно. Саме графічний інтерфейс забезпечує зв'язок між внутрішніми алгоритмами обробки зображень і практичним використанням програмного продукту.

Поєднання OpenCV та PyQt дозволяє створити комплексну програмну систему, у якій OpenCV виконує функції обробки та аналізу зображень, а PyQt відповідає за організацію візуальної взаємодії з користувачем. Такий підхід є зручним і ефективним, оскільки забезпечує логічний розподіл функцій між бібліотеками: одна відповідає за алгоритмічну складову, інша — за прикладну

реалізацію інтерфейсу. У результаті розробник отримує можливість створити не лише працездатну систему розпізнавання, а й програмний продукт, придатний для практичного застосування (табл. 1.1).

Таблиця 1.1

Порівняльна характеристика бібліотек OpenCV та PyQt

Характеристика	**OpenCV**	**PyQt**
Призначення	Обробка зображень та відео	Створення графічного інтерфейсу
Основні функції	Фільтрація, контури, Canny, threshold	Вікна, кнопки, GUI
Тип задач	Комп’ютерний зір	Інтерфейс користувача
Переваги	Висока швидкість, готові алгоритми	Зручний UI, інтерактивність
Використання у роботі	Розпізнавання фігур	Відображення результатів

Бібліотеки OpenCV та PyQt мають широкі функціональні можливості, що робить їх доцільним вибором для створення програмних систем комп’ютерного зору. OpenCV забезпечує ефективну реалізацію методів обробки та аналізу зображень, тоді як PyQt надає засоби для побудови зручного інтерфейсу користувача (рис. 1.5). Їх спільне використання дозволяє розробляти сучасні програмні застосунки, здатні виконувати розпізнавання об’єктів у режимі реального часу та забезпечувати наочне представлення результатів роботи системи.

Рис. 1.5 – Взаємодія бібліотек OpenCV та PyQt у програмній системі

Наведена схема демонструє логічну взаємодію між основними компонентами програмної системи комп’ютерного зору. Бібліотека OpenCV виконує ключову роль у процесі обробки та аналізу зображень, забезпечуючи реалізацію алгоритмів розпізнавання об’єктів, тоді як PyQt відповідає за візуалізацію отриманих результатів та організацію зручного інтерфейсу користувача.

Завдяки такому розподілу функцій досягається ефективна інтеграція алгоритмічної та прикладної частин системи, що дозволяє реалізувати повноцінний програмний застосунок для обробки зображень у режимі реального часу. Це забезпечує не лише коректне виконання задач розпізнавання, але й зручність взаємодії користувача з програмою [7].

Використання бібліотек OpenCV та PyQt є обґрунтованим і доцільним вибором для створення систем комп’ютерного зору, оскільки вони дозволяють поєднати ефективні алгоритми обробки зображень із сучасними засобами побудови графічного інтерфейсу, що є важливим для практичного застосування розробленої системи.

Висновок до розділу 1

Розглянуто теоретичні основи обробки та розпізнавання зображень у системах комп’ютерного зору. Проведений аналіз показав, що комп’ютерний зір є важливим напрямом сучасних інформаційних технологій, який забезпечує автоматизовану інтерпретацію візуальних даних та активно застосовується у різних сферах, зокрема у медицині, транспорті, системах безпеки та відеоспостереження.

Проаналізовано основні алгоритми та методи обробки зображень, що використовуються для розпізнавання об’єктів. Зокрема, розглянуто етапи попередньої обробки зображень, методи виділення меж, порогову обробку та алгоритми пошуку контурів. Встановлено, що поєднання цих методів дозволяє ефективно визначати геометричні характеристики об’єктів та здійснювати їх класифікацію.

Проаналізовано можливості бібліотек OpenCV та PyQt, які є доцільними для створення програмних систем комп’ютерного зору. OpenCV забезпечує реалізацію алгоритмів обробки та аналізу зображень, тоді як PyQt дозволяє створювати зручний графічний інтерфейс користувача. Їх поєднання дає змогу реалізувати повноцінний програмний застосунок, здатний працювати у режимі реального часу.

Результати проведеного аналізу підтверджують можливість та доцільність використання розглянутих методів і програмних засобів для розроблення системи розпізнавання зображень, що є основою для подальшого проєктування та реалізації програмного застосунку у наступних розділах роботи.

2 АНАЛІЗ ТА ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ГЕОМЕТРИЧНИХ ОБ'ЄКТІВ

2.1 Функціональна структура та модульна архітектура програмного застосунку

Функціональна структура програмного застосунку для розпізнавання зображень побудована за модульним принципом, що забезпечує гнучкість, масштабованість та зручність подальшого розширення системи. Такий підхід дозволяє розділити загальну логіку роботи програми на окремі функціональні компоненти, кожен з яких виконує чітко визначене завдання. Це спрощує процес розробки, тестування та підтримки програмного забезпечення (рис. 2.1).

Рис. 2.1 – Функціональна структура програмного застосунку

У структурі застосунку виділено кілька основних модулів, що взаємодіють між собою та формують єдину систему обробки та аналізу відеоданих (табл. 2.1).

Таблиця 2.1

Опис модулів системи

Модуль	**Функції**
Захоплення відео	Отримання кадрів
Обробка	Фільтрація, Canny
Розпізнавання	Контури, класифікація
GUI	Відображення

Першим етапом роботи системи є модуль захоплення відео, який відповідає за отримання вхідних даних. Даний модуль забезпечує підключення до камери або відеопотоку, зчитування кадрів у режимі реального часу та передачу їх до наступного етапу обробки. Важливою характеристикою цього модуля є стабільність роботи та швидкість обробки, оскільки від нього залежить безперервність функціонування всієї системи.

Наступним компонентом є модуль обробки зображення, який виконує попередню підготовку отриманих кадрів. У межах цього модуля здійснюється перетворення зображення у відтінки сірого, згладжування (blur), виділення меж об’єктів (наприклад, за допомогою алгоритму Canny) та порогова обробка. Основною метою цього етапу є підвищення якості зображення та виділення ключових ознак, необхідних для подальшого аналізу [8].

Після попередньої обробки дані передаються до модуля розпізнавання, який виконує аналіз об’єктів на зображенні. У цьому модулі здійснюється пошук контурів, визначення геометричних характеристик об’єктів (площа, периметр, кількість вершин) та їх класифікація. Саме цей модуль є центральним у системі, оскільки забезпечує виконання основної функції застосунку — розпізнавання об’єктів.

Окрему роль у структурі програмного забезпечення відіграє графічний інтерфейс користувача (GUI), реалізований із використанням бібліотеки PyQt (рис.

2.2). Даний модуль забезпечує взаємодію користувача із системою, дозволяючи запускати відеопотік, переглядати оброблені зображення, отримувати результати розпізнавання та керувати основними параметрами роботи програми. Наявність зручного інтерфейсу значно підвищує ефективність використання застосунку.

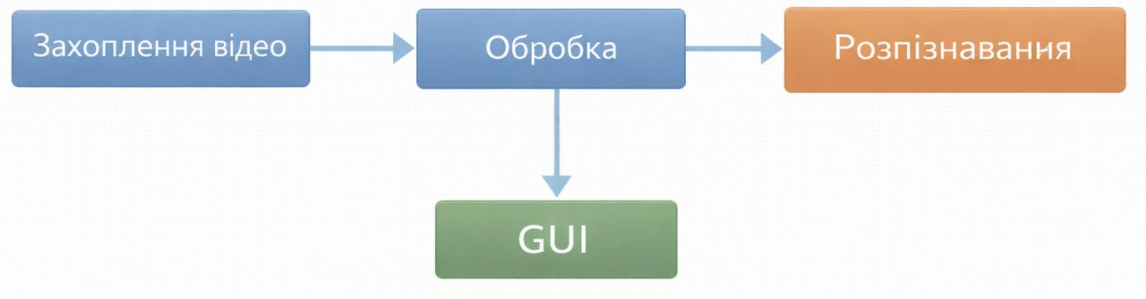


Рис. 2.2 – Модульна архітектура системи

Таким чином, модульна архітектура програмного застосунку дозволяє чітко розмежувати функціональні обов’язки між окремими компонентами системи. Взаємодія між модулями забезпечує послідовне проходження даних від етапу захоплення до отримання кінцевого результату. Такий підхід є ефективним для реалізації систем комп’ютерного зору та створює основу для подальшого розвитку і вдосконалення програмного продукту.

2.2 Алгоритмічна модель та принципи визначення геометричних фігур

Процедура розпізнавання геометричних об’єктів у розробленому застосунку базується на поєднанні класичних методів комп’ютерного зору, реалізованих із використанням бібліотеки OpenCV. Загальний підхід передбачає послідовну обробку вхідного зображення: від його попередньої підготовки та виділення контурів до подальшого визначення типу об’єкта на основі геометричних характеристик. Така поетапна схема дозволяє стабільно ідентифікувати фігури як у штучно сформованих сценах, так і у відеопотоці, навіть за наявності шумів, змін освітлення або незначних деформацій [17, 18].

Основні етапи обробки включають:

Перетворення зображення у відтінки сірого

Зображення, отримане з камери або згенерованого середовища, спочатку переводиться у градації сірого за допомогою функції cv2.cvtColor. Це дозволяє зменшити обсяг оброблюваних даних та підвищити ефективність подальших алгоритмів сегментації й аналізу.

Бінаризація (порогова обробка)

Після цього застосовується порогове перетворення (cv2.threshold або adaptiveThreshold), в результаті якого формується двійкове зображення. Об’єкти інтересу відображаються у вигляді світлих областей на темному фоні. Цей етап відіграє ключову роль, оскільки створює основу для точного виділення окремих фігур.

Пошук контурів

Далі використовується функція cv2.findContours, яка дозволяє виявити замкнуті області на зображенні. Кожен знайдений контур розглядається як окремий кандидат на геометричну фігуру.

Апроксимація контурів

З метою спрощення форми контурів застосовується алгоритм Дугласа-Пекера (cv2.approxPolyDP). Він зменшує кількість точок у контурі та дозволяє визначити кількість його вершин [20], що є базовою ознакою для подальшої класифікації:

- 3 вершини - трикутник;
- 4 вершини - квадрат або прямокутник;
- 5 - п’ятикутник;
- більше 5 - коло або складні багатокутники.

Обчислення геометричних характеристик

Для кожного знайденого об’єкта додатково визначаються його параметри:

- площа - за допомогою cv2.contourArea;
- довжина контуру (периметр) - cv2.arcLength;
- центр мас - на основі моментів (cv2.moments);
- габаритний прямокутник - cv2.boundingRect.

Класифікація об’єктів

Тип фігури встановлюється з урахуванням кількості вершин та пропорцій сторін. Наприклад, для чотирикутників використовується співвідношення ширини до висоти: якщо воно знаходиться в межах 0,9–1,1 — фігура визначається як квадрат, в іншому випадку — як прямокутник.

Оцінка ступеня округлості

Для фігур із великою кількістю вершин додатково обчислюється коефіцієнт округлості за формулою:

$$C = \frac{4\pi * S}{P^2} \tag{2.1}$$

де S — площа, P — периметр. Значення, близьке до 1, вказує на те, що об’єкт має форму, наближену до кола.

Формування результатів обробки

На завершальному етапі для кожної розпізнаної фігури формується набір даних, який включає її тип, геометричні характеристики, координати центру, розміри обмежувального прямокутника та, за необхідності, показник округлості. Отримана інформація передається до користувацького інтерфейсу для подальшого відображення.

Приклад реалізації алгоритму класифікації наведено на рис. 2.3.

```
if num_vertices == 3:
shape_type_ukrainian = "Трикутник"
elif num_vertices == 4:
if 0.9 <= aspect_ratio <= 1.1:
else:
shape_type_ukrainian = "Квадрат"
shape_type_ukrainian = "Прямокутник"
elif num_vertices == 5:
shape_type_ukrainian = "П''ятикутник"
elif num_vertices > 5:
if 0.80 < circularity < 1.20:
else:
shape_type_ukrainian = "Коло"
shape_type_ukrainian = "багатокутник"
```

Рис. 2.3 – Приклад коду для визначення типу фігури на основі кількості вершин

Слід підкреслити, що розроблений алгоритм спирається не лише на базові геометричні характеристики, але й враховує можливі відхилення форми об’єктів, зокрема перекоси або часткові спотворення контурів [20]. Завдяки цьому забезпечується стабільне функціонування системи навіть за складних умов зйомки, таких як нерівномірне освітлення чи наявність шумів. Крім того, модульна структура рішення передбачає можливість подальшого розширення, зокрема шляхом інтеграції методів глибинного навчання для розпізнавання більш складних об’єктів.

Для реалізації процесу ідентифікації геометричних фігур використано послідовний алгоритм, побудований у вигляді чітко визначених етапів - від отримання вхідного зображення до формування результатів класифікації. У процесі роботи враховуються ключові характеристики фігур, зокрема кількість вершин, пропорції сторін та коефіцієнт округлості, що забезпечує достатній рівень точності визначення типу об’єкта. На рис. 2.4 наведено блок-схему, яка відображає загальну логіку функціонування алгоритму в межах програмного застосунку.

Рис. 2.4 – Схематичне представлення алгоритму визначення геометричних фігур у системі комп’ютерного зору

Блок-схема ілюструє поетапний процес аналізу зображення, який включає перехід до відтінків сірого, виконання порогової обробки, виділення контурів, визначення кількості вершин, аналіз пропорцій сторін, розрахунок показника округлості та подальшу класифікацію об’єктів. Реалізація такого алгоритму забезпечує надійне розпізнавання основних геометричних фігур, зокрема трикутників, квадратів, прямокутників, кіл і багатокутників, на основі їх ключових геометричних характеристик.

2.3 Особливості потокової обробки відеоданих у системі розпізнавання

Однією з важливих характеристик сучасних систем комп'ютерного зору є можливість обробки відеоданих у режимі реального часу з підтримкою стабільної частоти кадрів, мінімальних затримок та безперервного відображення результатів. З цією метою у розробленому програмному продукті реалізовано механізм потокової обробки відео, який базується на використанні багатопотокового програмування засобами бібліотеки PyQt5 [21].

Обробка відеопотоку винесена в окремий програмний модуль — клас WebcamThread, що успадковує функціональність QThread. Такий підхід дозволяє відокремити процес отримання та обробки кадрів від основного потоку графічного інтерфейсу, що, у свою чергу, запобігає його зависанню під час виконання обчислень (рис. 2.5).

```
class WebcamThread(QThread):
frame_ready = pyqtSignal(QPixmap)
def run(self):
self.cap = cv2.VideoCapture(self.camera_index)
while self.running:
ret, frame = self.cap.read()
if ret:
# Обробка кадру
```

Рис. 2.5 – Схема реалізації багатопотокової обробки відеопотоку

Використання механізму `pyqtSignal` забезпечує неблокуючу передачу даних між потоком, відповідальним за роботу з камерою, та графічним інтерфейсом. Завдяки цьому оновлення зображення у вікні відбувається без затримок і не потребує очікування завершення обробки кожного кадру.

Обробка відеокадрів у реальному часі

Кожен кадр, отриманий за допомогою `cv2.VideoCapture`, передається до основного модуля аналізу (`shape_detector.detect_shapes`) [22], де послідовно виконуються такі операції:

1. перетворення зображення у відтінки сірого;
2. застосування порогової обробки;

3. виявлення та аналіз контурів;
4. формування структури даних з інформацією про кожен об’єкт;
5. повернення обробленого кадру з візуальним виділенням і підписами фігур.

Отримані результати передаються через сигнал `processed_frame_ready` у вигляді об’єкта `QPixmap`, що забезпечує оперативне оновлення елементів інтерфейсу користувача [22].

Обробка помилок та стабільність роботи

Для підвищення надійності системи передбачено механізм обробки винятків. У випадку виникнення проблем із доступом до камери система коректно інформує користувача та зупиняє процес захоплення відео:

```
except Exception as e:
    print(f"Error initializing camera: {e}")
    self.cap = None
```

Продуктивність та швидкодія

Застосування багатопотокової обробки дає змогу підтримувати швидкість обробки на рівні приблизно 10–15 кадрів за секунду залежно від обчислювальних ресурсів системи. Цього достатньо для задач розпізнавання базових геометричних фігур. Відсутність операцій запису у файлову систему дозволяє мінімізувати затримки та забезпечує ефект обробки в режимі реального часу [23].

Взаємодія з інтерфейсом користувача

Графічний інтерфейс постійно оновлюється новими кадрами, тоді як інформаційна панель відображає перелік виявлених фігур. Це надає користувачеві можливість спостерігати результати обробки у динаміці, що особливо корисно під час тестування алгоритмів і налаштування їх параметрів.

Висновок до розділу 2

проаналізовано функціональну структуру та модульну архітектуру розробленого програмного застосунку. Визначено основні складові системи, їх призначення та взаємодію між окремими модулями, що забезпечує цілісність та гнучкість програмного рішення. Особливу увагу приділено принципам побудови архітектури, які дозволяють легко масштабувати систему та інтегрувати додаткові функціональні можливості.

Розглянуто алгоритмічну модель розпізнавання геометричних фігур, яка базується на використанні класичних методів комп’ютерного зору. Проаналізовано ключові етапи обробки зображення, включаючи попередню підготовку, виділення контурів, апроксимацію та класифікацію об’єктів за геометричними ознаками. Встановлено, що використання таких характеристик, як кількість вершин, пропорції сторін та коефіцієнт округлості, забезпечує достатню точність ідентифікації фігур.

Досліджено особливості реалізації потокової обробки відеоданих у системі розпізнавання. Розглянуто механізми багатопотокового програмування, що дозволяють ефективно обробляти відеопотік у реальному часі без блокування інтерфейсу користувача. Визначено, що застосування асинхронної передачі даних та оптимізація обробки кадрів забезпечують стабільну продуктивність системи та комфортну взаємодію користувача із застосунком.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

3.1 Архітектура програмної реалізації та структура системи

Розробка програмного застосунку для розпізнавання зображень виконувалась із застосуванням методів комп'ютерного зору, які забезпечують автоматизовану обробку, аналіз і подальшу інтерпретацію візуальних даних. Система побудована на основі модульного підходу, що сприяє її гнучкості, можливості масштабування та спрощує супровід і подальший розвиток. До складу програмного рішення входять модулі захоплення відеоданих, попередньої обробки кадрів, аналізу контурів, класифікації геометричних об'єктів, а також користувацький інтерфейс для взаємодії із системою. Для реалізації обрано мову програмування Python із використанням бібліотек OpenCV, NumPy та PyQt, що дозволяє ефективно працювати з зображеннями та забезпечує зручне графічне середовище. У цьому підрозділі детально розглядаються особливості реалізації ключових компонентів і загальна структура програмної системи [24].

Структура проєкту наведена на рисунку 3.1. У головній директорії SQUARE розміщені основні модулі застосунку, кожен із яких відповідає за окрему функціональну частину системи.

Зокрема, структура включає такі елементи:

– ***main.py*** - основний файл запуску, який ініціалізує компоненти системи та відповідає за старт графічного інтерфейсу;

– ***scene_generator.py*** - модуль, призначений для створення тестових сцен або синтетичних зображень із геометричними фігурами;

– ***shape_detector.py*** - ключовий модуль, що реалізує алгоритми розпізнавання фігур на основі методів комп'ютерного зору;

– ***utils.py*** - допоміжний компонент, який містить набір службових функцій для обробки даних та підтримки роботи інших модулів;

– ***webcam_thread.py*** - модуль, що забезпечує роботу з відеопотоком у окремому потоці, що дозволяє уникнути блокування інтерфейсу користувача;

– ***requirements.txt*** - файл, що містить перелік необхідних бібліотек для встановлення та коректної роботи застосунку.

Крім того, у структурі проєкту присутні каталоги ***.venv*** та ***.venv1***, які використовуються для ізоляції залежностей у межах віртуального середовища Python, що сприяє стабільності та незалежності роботи програмного продукту [25].

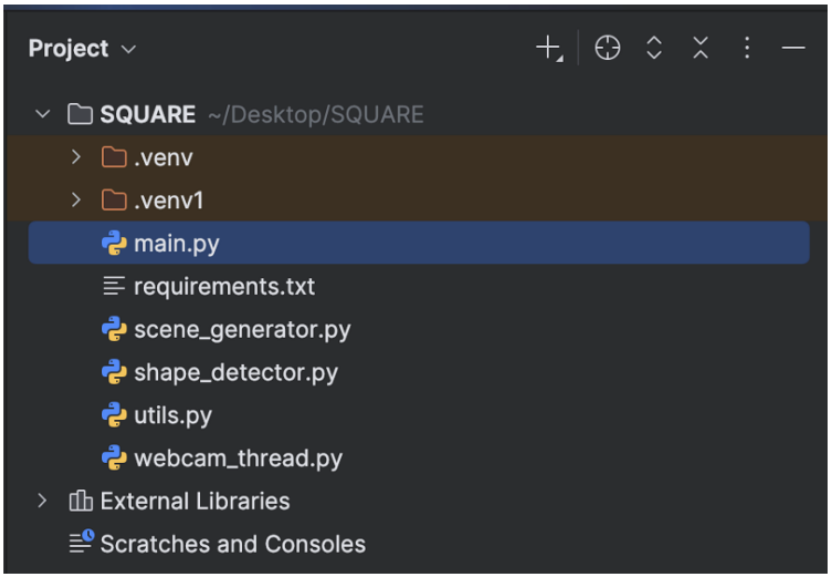


Рис. 3.1 – Архітектурна структура програмного проєкту системи розпізнавання зображень

Головний модуль **main.py** (рис. 3.2) відповідає за реалізацію графічного інтерфейсу користувача із використанням фреймворку PyQt5. У цьому файлі визначено основний клас **ShapeSenseApp,** який наслідує **QMainWindow** та виконує функції центрального керуючого елемента застосунку. Він забезпечує ініціалізацію інтерфейсу, взаємодію з модулем обробки відеопотоку та обробку подій користувача [25].

У конструкторі класу **(init)** реалізовано:

– задання назви головного вікна «Детектор фігур» та встановлення фіксованих розмірів 1250×750 пікселів;

- ініціалізацію змінних стану, зокрема **is_webcam_running, generated_canvas** та списку **ground_truth_shapes,** які використовуються для контролю роботи відеопотоку та збереження інформації про згенеровані об’єкти;
- створення та підключення екземпляра класу **WebcamThread,** що працює в окремому потоці та відповідає за захоплення відео.

Обмін даними між потоками організовано за допомогою сигнально-слотового механізму Qt, зокрема для обробки отриманих кадрів **(frame_ready),** бінаризованих зображень (**threshold_frame_ready**) та інформації про розпізнані фігури (**shapes_info_ready**);

Виклик допоміжних методів **initUI()** та **set_initial_pixmaps(),** які забезпечують налаштування зовнішнього вигляду інтерфейсу та встановлення початкових зображень.

У методі **initUI()** реалізовано формування структури інтерфейсу:

- створюється центральний віджет та основне компонування на базі **QVBoxLayout;**
- формується ліва панель керування із фіксованою шириною 320 пікселів;
- додається кнопка «Увімкнути вебкамеру», яка пов’язана зі слотом **toggle_webcam** та відповідає за запуск або зупинку відеопотоку;
- передбачено текстовий заголовок, що позначає блок керування камерою та покращує зручність використання інтерфейсу [25].

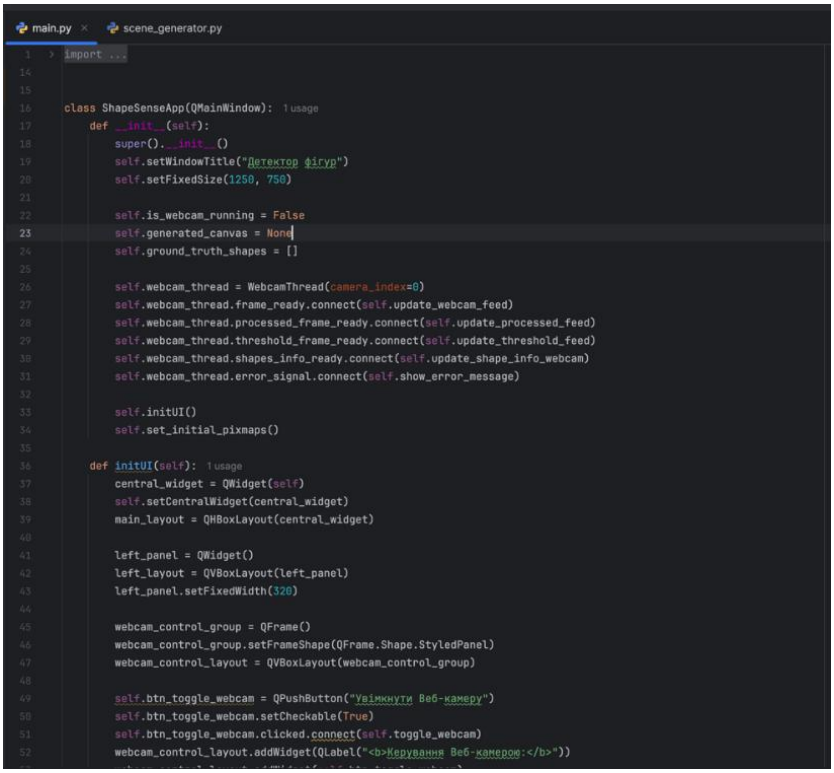


Рис. 3.2 – Частина коду main.py з класом ShapeSenseApp, що відповідає за запуск застосунку та роботу з відео

Модуль генерації сцен **scene_generator.py** (рис. 3.3) призначений для формування випадкових тестових зображень із геометричними фігурами, що використовуються для оцінювання ефективності алгоритмів розпізнавання. Основною функцією модуля є **generate_random_scene(width, height, shape_config),** яка створює зображення з випадковим розташуванням об’єктів на світлому фоні відповідно до заданих параметрів розміру, кількості та типів фігур.

На початковому етапі роботи функції:

формується біле зображення заданих розмірів *width* × *height* із типом даних *uint8*; задаються параметри генерації, зокрема мінімальний та максимальний розміри об’єктів (*min_size*, *max_size*), відступи від країв зображення (*border*) та мінімальна відстань між фігурами (*min_gap*);

ініціалізуються списки **ground_truth** і **existing_bboxes**, які використовуються для збереження інформації про створені об’єкти та контролю їх взаємного розташування.

Для уникнення накладання фігур застосовується допоміжна функція **bboxes_overlap(),** яка перевіряє перетин обмежувальних прямокутників (bounding box) з урахуванням заданого мінімального інтервалу між об’єктами [26].

На основі вхідного параметра **shape_config** формується список **all_shapes_to_generate,** що містить типи фігур та кількість екземплярів кожного типу. Після випадкового перемішування елементів забезпечується довільний порядок генерації об’єктів.

У процесі ітерації для кожної фігури:

– генерується унікальний ідентифікатор (*shape_id*);
– створюється структура даних **info,** яка містить службову інформацію про об’єкт (тип, ідентифікатор тощо);
– задається випадкове кольорове оформлення фігури.

Загалом, даний модуль виконує важливу функцію тестування системи, оскільки дозволяє створювати контрольовані сцени для подальшого аналізу точності розпізнавання та порівняння отриманих результатів.

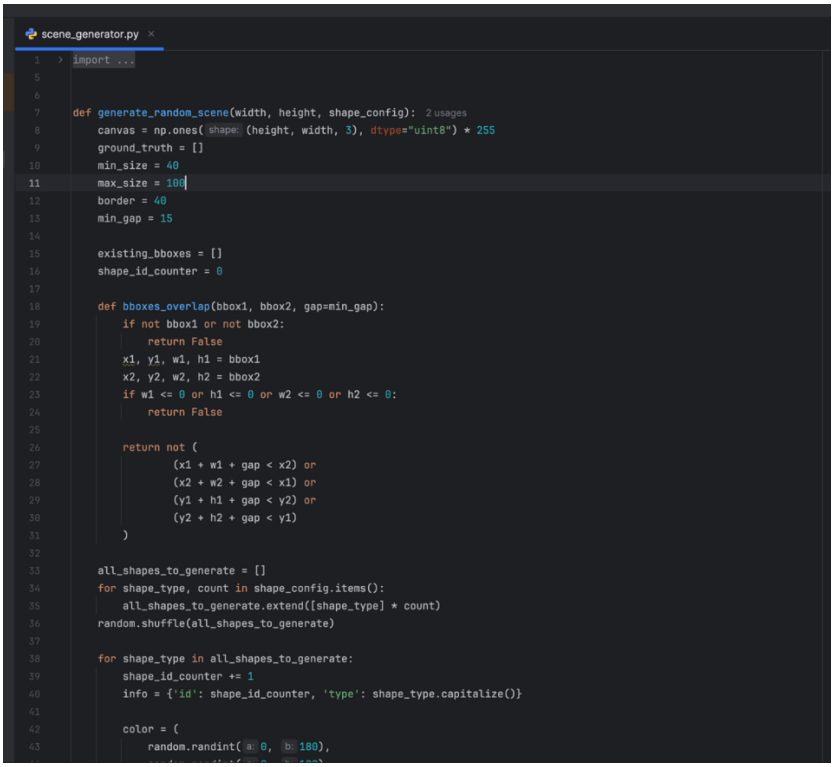


Рис. 3.3 – Реалізація генерації тестових зображень із геометричними фігурами у модулі scene_generator.py

Модуль розпізнавання геометричних об’єктів shape_detector.py (рис. 3.4) виконує функції виявлення та класифікації фігур на основі вхідного зображення, яким може бути як кадр із відеопотоку, так і згенерована сцена. Ключовим елементом модуля є функція detect_shapes(frame), яка реалізує поетапний алгоритм обробки зображення із застосуванням інструментів бібліотеки OpenCV [26].

Функція працює за наступною логікою:

1. виконується перетворення зображення у відтінки сірого та застосовується згладжування (GaussianBlur) з метою зменшення шумів;
2. здійснюється виявлення границь об’єктів за допомогою оператора Кенні (cv2.Canny);
3. проводиться дилатація отриманих контурів для підвищення їх чіткості;
4. виконується пошук контурів на зображенні із використанням функції cv2.findContours.

Під час обробки кожного знайденого контуру:

– обчислюється площа, і якщо її значення менше 150 пікселів, такий контур відкидається як шум;

– визначається довжина контуру (arcLength) та виконується його спрощення методом апроксимації (approxPolyDP);

– встановлюється кількість вершин (num_vertices), яка використовується як основний критерій для класифікації геометричних фігур (наприклад, трикутник, прямокутник, коло тощо);

– обчислюються геометричні моменти, на основі яких визначається центр мас фігури (cX, cY);

– додатково розраховується співвідношення сторін (aspect_ratio), що дає змогу розрізняти квадрат і прямокутник.

Для об’єктів, які не вдалося однозначно класифікувати, за замовчуванням встановлюється тип «Невідома» (у двох варіантах — українською та латиницею).

Для кожної фігури формується структура даних shape_info, що містить основні характеристики об’єкта, зокрема його тип, координати, колір та інші параметри [26].

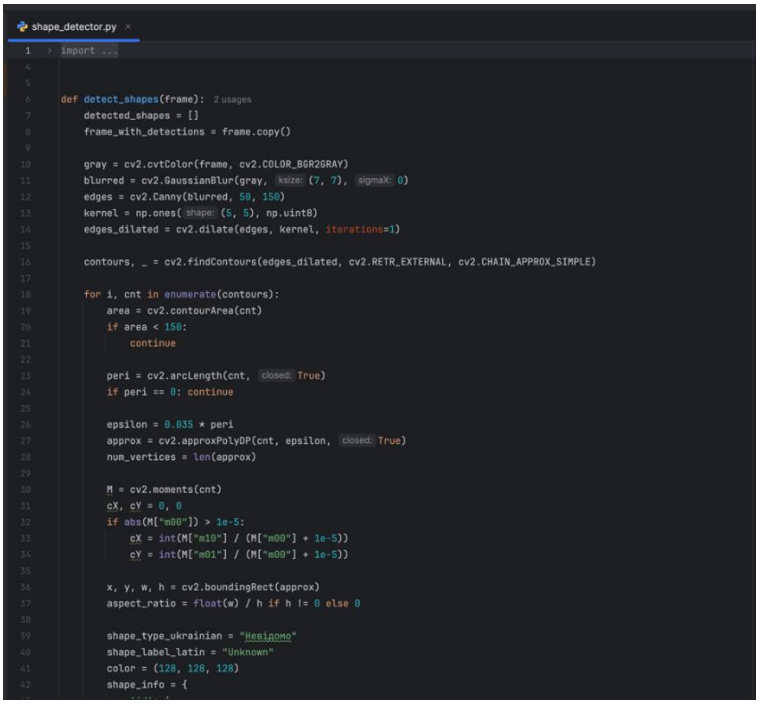


Рис. 3.4 – Частина коду shape_detector.py для визначення геометричних фігур на зображенні

Допоміжний модуль utils.py (рис. 3.5) призначений для реалізації сервісних функцій, які використовуються в різних компонентах програмного застосунку з метою спрощення роботи із зображеннями та результатами їх обробки. Даний модуль не виконує безпосереднього розпізнавання або аналізу об’єктів, проте забезпечує підтримку процесів відображення та представлення отриманих даних у зручному вигляді.

У межах модуля реалізовано дві ключові функції:

- convert_cv_qt(cv_img) - функція, яка здійснює перетворення зображення з формату OpenCV (масив NumPy) у формат QPixmap, що дозволяє відобразити його у графічному інтерфейсі, побудованому на базі Qt. У процесі виконання:
- відбувається конвертація кольорової моделі з BGR у RGB;
- визначаються параметри зображення, зокрема розміри та кількість байтів на рядок;
- створюється об’єкт QImage, який надалі перетворюється у QPixmap;

– у випадку виникнення помилки здійснюється її виведення в консоль, а функція повертає порожній результат.

format_shape_info(shape_data) - функція, що формує структурований текстовий опис характеристик розпізнаного об’єкта. Зокрема, вона включає:

– тип геометричної фігури; – координати її центру;

– значення площі та периметра;

– параметри обмежувального прямокутника (bounding box);

– коефіцієнт округлості (circularity).

Сформований текст подається у зрозумілому та впорядкованому вигляді, що забезпечує його ефективне використання, наприклад, для відображення у вікні результатів розпізнавання [27].

shape_detector.py utils.py

```python
import ...

def convert_cv_qt(cv_img):  7 usages
    try:
        rgb_image = cv2.cvtColor(cv_img, cv2.COLOR_BGR2RGB)
        h, w, ch = rgb_image.shape
        bytes_per_line = ch * w
        qt_image = QImage(rgb_image.data, w, h, bytes_per_line, QImage.Format.Format_RGB888)
        return QPixmap.fromImage(qt_image)
    except Exception as e:
        print(f"Error converting cv image to qt: {e}")
        return QPixmap()


def format_shape_info(shape_data):  3 usages
    info = f"Тип: {shape_data.get('type', 'N/A')}\n"
    center = shape_data.get('center', None)
    if center:
        info += f"  Центр: ({center[0]}, {center[1]})\n"
    info += f"  Площа: {shape_data.get('area', 0):.1f}\n"
    info += f"  Периметр: {shape_data.get('perimeter', 0):.1f}\n"
    bbox = shape_data.get('bbox', None)
    if bbox:
        info += f"  Рамка: x={bbox[0]}, y={bbox[1]}, w={bbox[2]}, h={bbox[3]}\n"
    circ = shape_data.get('circularity', None)
    if circ is not None:
        info += f"  Округлість: {circ:.3f}\n"
    info += "-" * 20 + "\n"
    return info
```

Рис. 3.5 – Реалізація допоміжного модуля utils.py для обробки зображень і представлення результатів розпізнавання

Модуль обробки відеопотоку webcam_thread.py (рис. 3.6) забезпечує асинхронне захоплення кадрів із вебкамери з використанням окремого потоку виконання, що дозволяє уникнути блокування графічного інтерфейсу користувача.

Реалізація побудована на базі класу WebcamThread, який успадковує функціональність QThread із бібліотеки PyQt [27, 28].

У межах класу визначено набір сигналів, що використовуються для обміну даними між потоками:

– frame_ready, processed_frame_ready, threshold_frame_ready - передають кадри на різних етапах обробки (оригінальний, оброблений та бінаризований);

– shapes_info_ready - використовується для передачі інформації про розпізнані геометричні об’єкти;

– error_signal - призначений для інформування про помилки, що виникають під час ініціалізації або роботи з відеопристроєм.

У конструкторі класу (init()) виконується:

– налаштування параметрів доступу до камери за її індексом;

– ініціалізація початкових значень змінних, зокрема running = False та cap = None.

Основний функціонал реалізовано в методі run(), який визначає логіку роботи потоку:

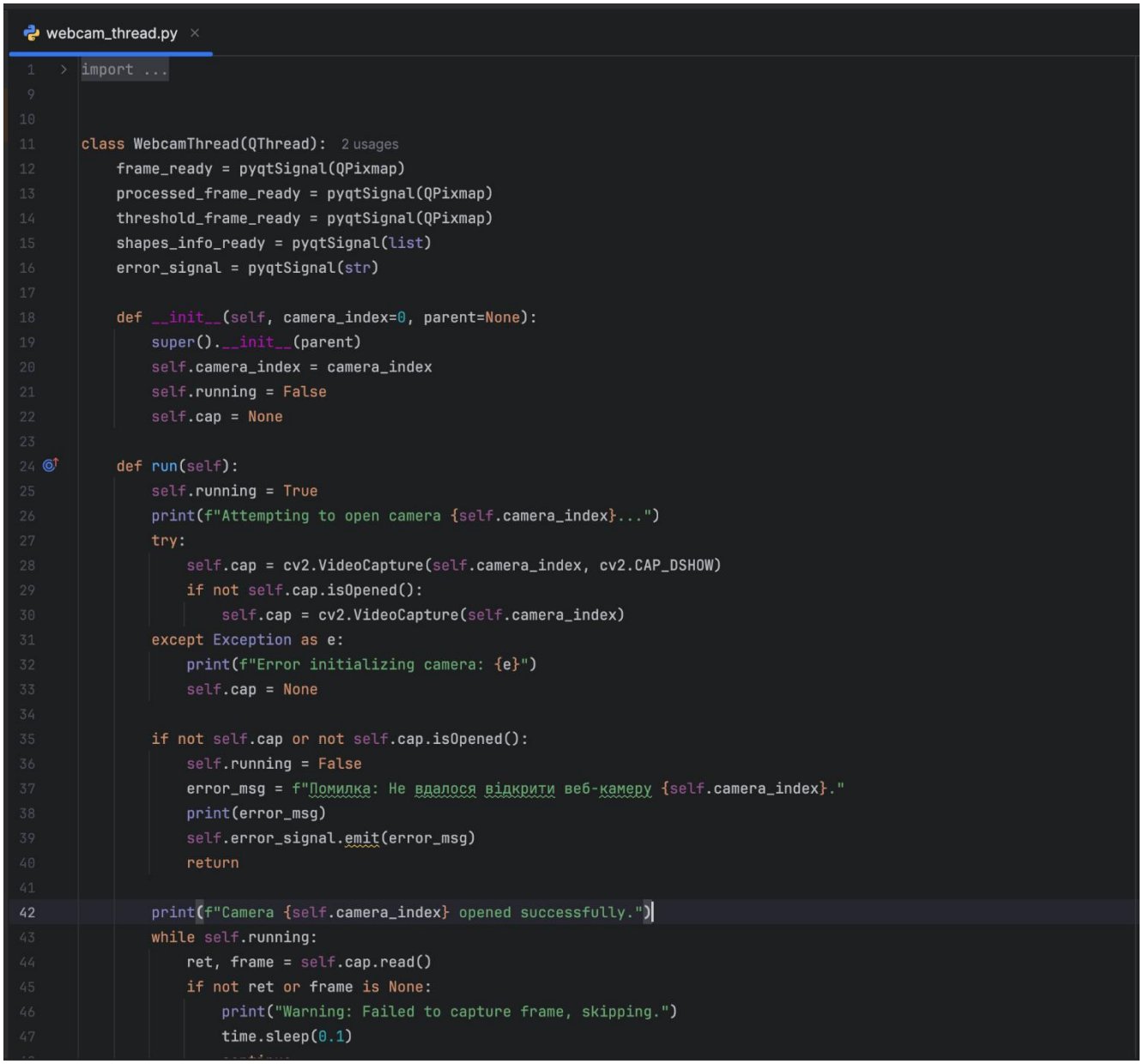
– встановлюється прапорець self.running = True та виконується спроба підключення до камери через cv2.VideoCapture;

– у разі успішної ініціалізації виводиться відповідне повідомлення у консоль;

– якщо доступ до камери отримати не вдалося - генерується повідомлення про помилку через сигнал error_signal;

– у циклі while self.running відбувається зчитування кадрів методом read(). Якщо отримати кадр не вдалося, ітерація пропускається, після чого виконується повторна спроба з невеликою затримкою.

Таким чином, даний модуль відіграє важливу роль у забезпеченні безперервного відеопотоку та його подальшої обробки, не створюючи додаткового навантаження на основний потік виконання програми.



```python
import ...


class WebcamThread(QThread):  2 usages
    frame_ready = pyqtSignal(QPixmap)
    processed_frame_ready = pyqtSignal(QPixmap)
    threshold_frame_ready = pyqtSignal(QPixmap)
    shapes_info_ready = pyqtSignal(list)
    error_signal = pyqtSignal(str)

    def __init__(self, camera_index=0, parent=None):
        super().__init__(parent)
        self.camera_index = camera_index
        self.running = False
        self.cap = None

    def run(self):
        self.running = True
        print(f"Attempting to open camera {self.camera_index}...")
        try:
            self.cap = cv2.VideoCapture(self.camera_index, cv2.CAP_DSHOW)
            if not self.cap.isOpened():
                self.cap = cv2.VideoCapture(self.camera_index)
        except Exception as e:
            print(f"Error initializing camera: {e}")
            self.cap = None

        if not self.cap or not self.cap.isOpened():
            self.running = False
            error_msg = f"Помилка: Не вдалося відкрити веб-камеру {self.camera_index}."
            print(error_msg)
            self.error_signal.emit(error_msg)
            return

        print(f"Camera {self.camera_index} opened successfully.")
        while self.running:
            ret, frame = self.cap.read()
            if not ret or frame is None:
                print("Warning: Failed to capture frame, skipping.")
                time.sleep(0.1)
```

Рис. 3.6 – Частина модуля webcam_thread.py для зчитування відеопотоку в окремому потоці

На рисунку 3.7 представлено структуру ключових класів, що забезпечують функціонування програмного застосунку. Центральним елементом є клас ShapeSenseApp, який реалізує головне вікно та координує взаємодію з іншими компонентами системи, зокрема WebcamThread, shape_detector, scene_generator, а також використовує допоміжні функції з модуля utils.py. Відображені зв’язки між класами ілюструють процес обміну даними, виклики методів та обробку сигналів у межах використання фреймворку PyQt [28].

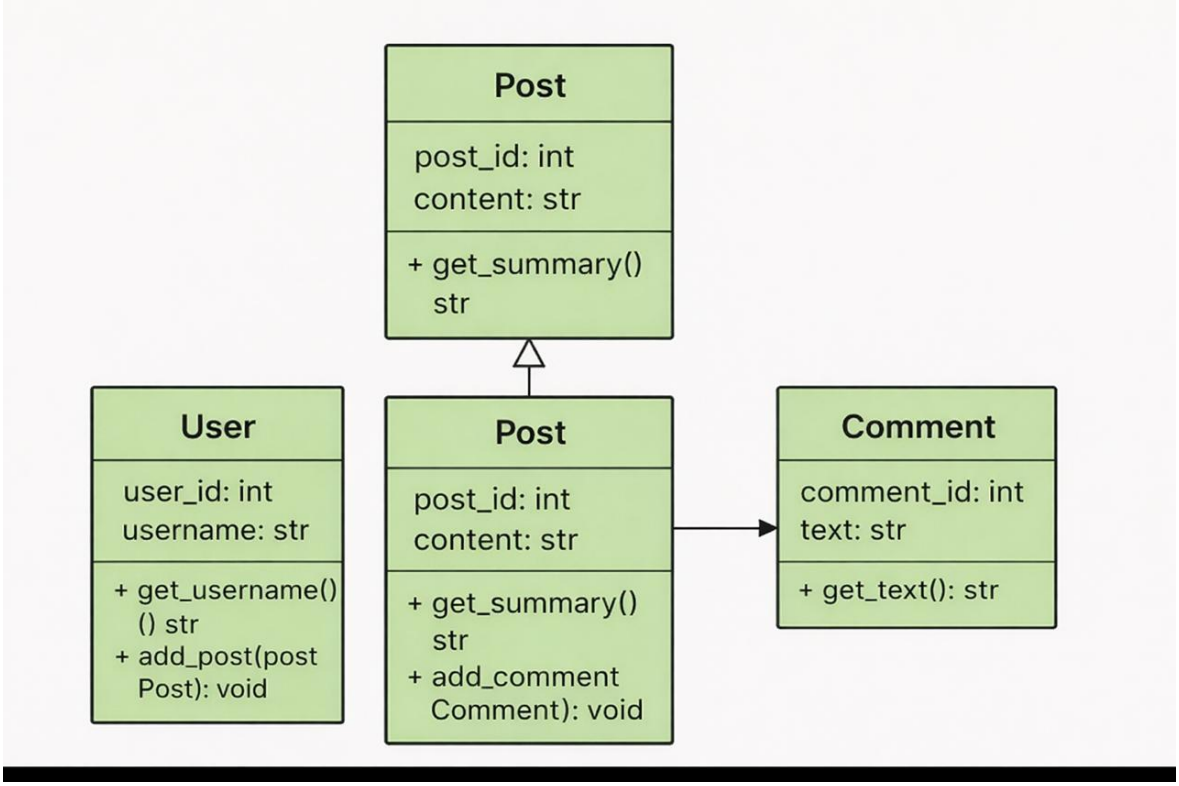


Рис. 3.7 – UML-діаграма класів системи розпізнавання зображень

3.2 Розроблення інтерфейсу користувача та режими функціонування застосунку

Графічний інтерфейс розробленого програмного застосунку реалізовано із використанням бібліотеки PyQt6, що забезпечує зручну та зрозумілу взаємодію користувача з основними функціями системи. Інтерфейс орієнтований на керування режимами роботи (використання вебкамери або генерація тестових сцен), відображення результатів обробки та виведення текстової інформації про розпізнані об’єкти. Його структура побудована у вигляді окремих функціональних зон: панелі керування, інформаційного блоку та області відображення зображення, що сприяє комфортному використанню навіть для користувачів без спеціальної підготовки. У даному підрозділі детально розглядаються принципи організації інтерфейсу, його основні елементи та особливості роботи в різних режимах [33].

Одним із ключових компонентів є панель керування, розміщена у лівій частині головного вікна, яка відповідає за взаємодію з вебкамерою та модулем

генерації сцен. Вона реалізує дві основні групи функцій.

Керування вебкамерою.

За допомогою кнопки «Увімкнути вебкамеру» користувач може запускати або зупиняти процес відеозахоплення. Під час роботи застосунку назва кнопки автоматично змінюється на «Вимкнути вебкамеру», що забезпечує зручне двостороннє керування цим режимом.

Генерація тестових сцен.

Інтерфейс надає можливість задавати кількість геометричних фігур для генерації за допомогою спеціальних елементів керування (лічильників), зокрема:

- кола;
- прямокутники;
- трикутники.

Після встановлення необхідних параметрів натискання кнопки «Генерувати сцену та розпізнати» ініціює створення випадкового зображення з відповідними фігурами та автоматичний запуск алгоритму їх розпізнавання [29].

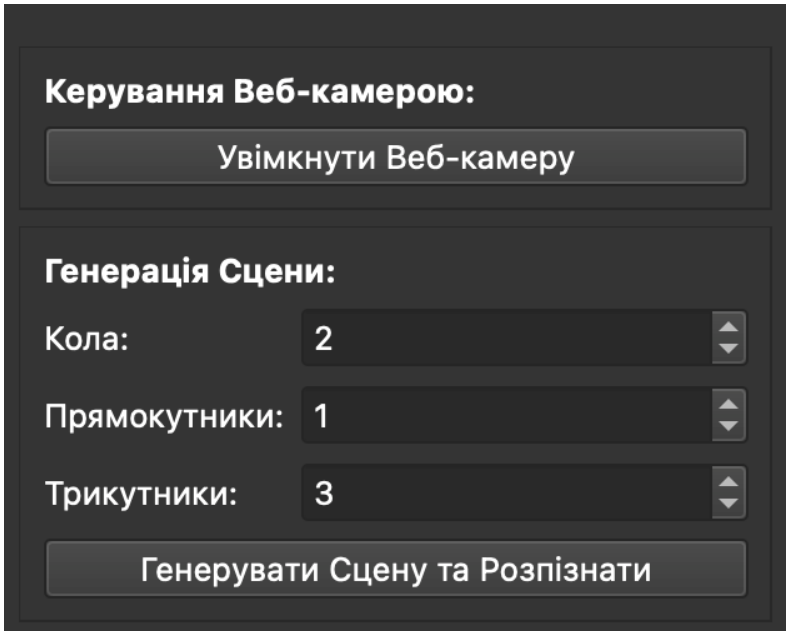


Рис. 3.8 – Елемент інтерфейсу для налаштування відеозахоплення та генерації геометричних сцен

Іншим важливим компонентом лівої частини інтерфейсу є інформаційна панель, призначена для відображення детальних характеристик розпізнаних об’єктів. Після обробки зображення - незалежно від джерела (відеопотік або

згенерована сцена) - у цьому текстовому полі відображається інформація про кожну фігуру, зокрема її тип, координати центру, площа, периметр, параметри обмежувального прямокутника (bounding box), а також коефіцієнт округлості (за наявності відповідного обчислення).

Даний елемент інтерфейсу реалізовано з використанням віджета QTextEdit, для якого встановлено моноширинний шрифт та вимкнено автоматичне перенесення рядків. Це дозволяє забезпечити чітке та впорядковане відображення структурованих даних, підвищуючи зручність їх сприйняття.

Інформаційна панель виконує роль аналітичного блоку, надаючи користувачу розширені відомості про геометричні параметри виявлених об’єктів у межах оброблюваної сцени або відеопотоку [29].

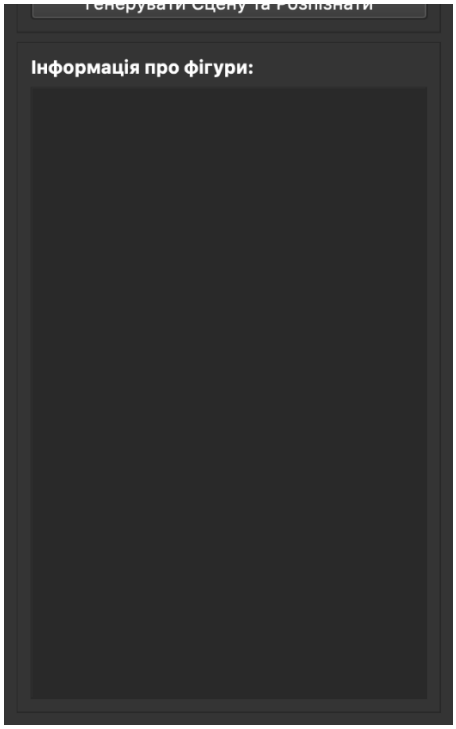


Рис. 3.9 – Інтерфейс відображення параметрів розпізнаних фігур у текстовому блоці

У правій частині інтерфейсу розміщена область візуалізації, призначена для відображення відеопотоку та результатів його обробки. Вона структурована на три функціональні зони, кожна з яких виконує окрему роль у процесі аналізу зображення.

Верхня частина, позначена як «Веб-камера (Результат)», використовується для відображення обробленого відео, на якому вже нанесено результати

розпізнавання геометричних фігур. Саме в цьому блоці користувач бачить підсумковий результат роботи алгоритмів комп’ютерного зору.

Нижня ліва зона (Raw) відображає оригінальний відеопотік без будь-якої обробки. Це дає змогу оцінити якість вхідного зображення та виявити можливі проблеми, пов’язані з освітленням або шумами.

Нижня права зона (Thresh) демонструє зображення після попередньої обробки, зокрема після застосування операторів виділення контурів (наприклад, cv2.Canny). Це дозволяє наочно побачити, як формується основа для подальшого аналізу об’єктів.

Загалом така організація інтерфейсу забезпечує можливість одночасного контролю всіх етапів обробки від вхідного сигналу до фінального результату, що значно спрощує процес тестування, аналізу та налагодження роботи системи [30].

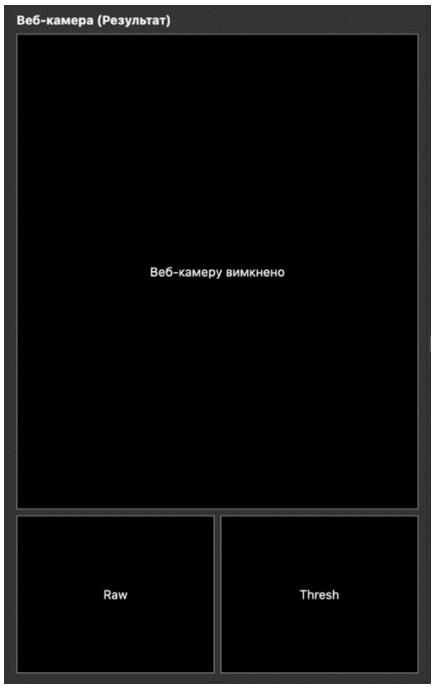


Рис. 3.10 – Інтерфейс візуалізації відеоданих: сирий потік, оброблене (threshold) зображення та результати детекції

Додатковим важливим елементом графічного інтерфейсу є зона генерації сцени, яка розташована у правій частині вікна поруч із блоком відображення відеопотоку. Даний компонент призначений для візуалізації синтетично сформованих зображень, а також результатів їх подальшого аналізу за допомогою

алгоритмів комп’ютерного зору.

У верхній частині цього блоку відображається оброблене зображення, на якому виділено контури геометричних фігур і нанесено відповідні підписи. Це дозволяє оцінити коректність роботи алгоритму розпізнавання.

У нижній частині (яка може не бути представлена на рисунку) відображається вихідне згенероване зображення без обробки. Воно використовується як еталон для порівняння результатів детекції та оцінки точності виявлення об’єктів.

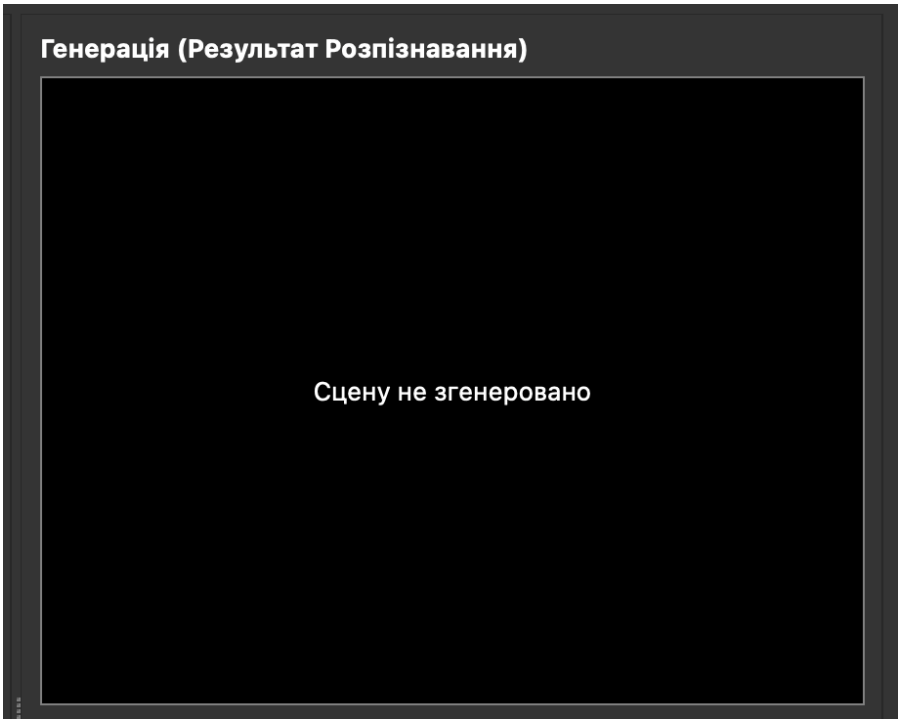


Рис. 3.11 – Візуалізація результатів детекції об’єктів на синтетично сформованому зображенні

У нижній частині правої панелі інтерфейсу передбачено окрему область для відображення вихідного згенерованого зображення. Даний елемент дозволяє користувачеві переглянути сцену з геометричними фігурами у первинному вигляді, тобто до виконання будь-яких операцій аналізу та розпізнавання. Це створює можливість наочного порівняння початкового зображення з результатами обробки, що є важливим для оцінювання точності роботи алгоритму.

Зображення відображається без додаткових графічних елементів - контурів, підписів або маркерів - що забезпечує об’єктивність сприйняття вихідних даних та

дозволяє коректно зіставляти їх із результатами розпізнавання.

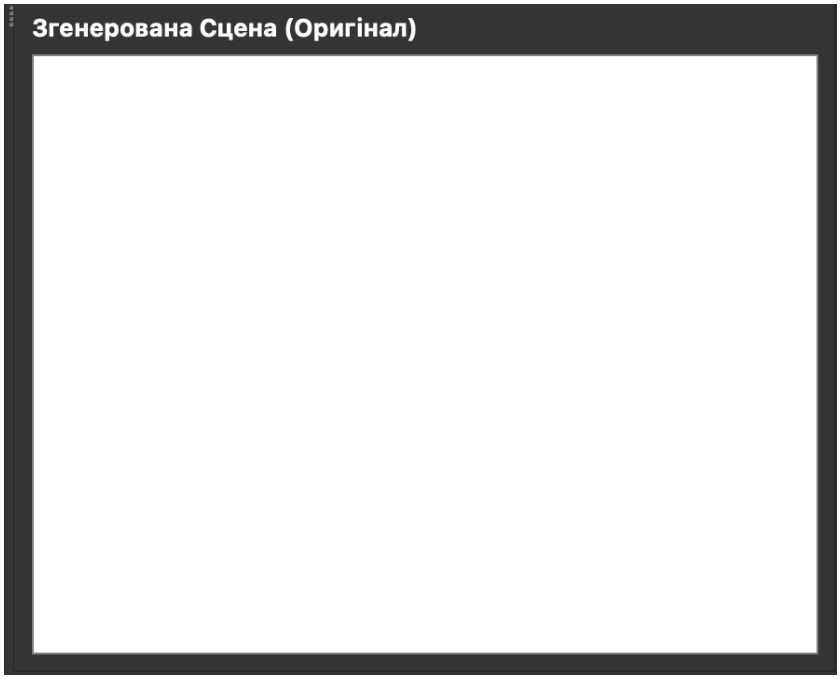


Рис. 3.12 – Виведення початкового синтетичного зображення без обробки для оцінки результатів розпізнавання

У межах даного підрозділу проаналізовано принципи побудови користувацького інтерфейсу програмного застосунку для розпізнавання геометричних фігур. Інтерфейс реалізовано із використанням бібліотеки PyQt6 та організовано за принципом поділу на функціональні області: керування вебкамерою, модуль генерації сцен, зона відображення результатів обробки та блок текстового представлення характеристик об’єктів. Така організація забезпечує зручну та зрозумілу взаємодію користувача із системою, що є доцільним як для тестування алгоритмів, так і для навчального або демонстраційного використання [30].

Реалізований інтерфейс дозволяє виконувати такі основні дії:

- запуск і зупинку відеопотоку з вебкамери;
- формування користувацьких сцен із заданими параметрами геометричних фігур;
- одночасне відображення вихідного та обробленого зображення;
- отримання структурованої текстової інформації про характеристики кожного

виявленого об’єкта.

Завдяки модульній організації та чіткій структурі інтерфейс легко піддається модифікації, що відкриває можливості для його адаптації до інших задач у сфері комп’ютерного зору, а також для розширення функціональності шляхом додавання нових режимів роботи.

3.3 Експериментальне тестування системи в умовах різного освітлення та обмежених обчислювальних ресурсів

Розроблений програмний застосунок передбачає два основних режими роботи, які забезпечують можливість як обробки даних у реальному часі, так і проведення контрольованого тестування алгоритмів комп’ютерного зору.

1. Режим розпізнавання у реальному часі (вебкамера).

У цьому режимі здійснюється безперервне захоплення відеопотоку з вебкамери з подальшою обробкою кожного кадру та визначенням геометричних об’єктів.

Результати обробки відображаються у декількох форматах:

- оброблене відео із нанесеними контурами та підписами фігур;
- необроблений відеопотік (Raw), що використовується для порівняння;
- зображення після попередньої обробки (Thresh), яке демонструє результат
- застосування алгоритмів виділення контурів (наприклад, оператора Кенні).

Застосування цього режиму дозволяє оцінити роботу алгоритму в умовах, наближених до реальних, з урахуванням змін освітлення, наявності шумів та сторонніх об’єктів. Це дає змогу аналізувати продуктивність системи, швидкість обробки та її стійкість до впливу зовнішніх факторів.

2. Режим розпізнавання на згенерованій сцені.

Даний режим призначений для проведення контрольованого тестування алгоритму. Користувач має можливість задати кількість і типи геометричних фігур (зокрема кола, прямокутники та трикутники), після чого система формує сцену з випадковим розташуванням об’єктів. Подальший аналіз здійснюється на основі

цього зображення, що дозволяє:

- зіставляти результати розпізнавання з еталонними даними (ground truth);
- оцінювати точність і надійність алгоритму в умовах відсутності випадкових впливів;
- використовувати згенеровані сцени для навчальних цілей, демонстрацій або тестування.

Організація інтерфейсу передбачає окремі області для відображення як вихідного зображення, так і результатів його обробки, що створює зручні умови для аналізу ефективності роботи системи [31].

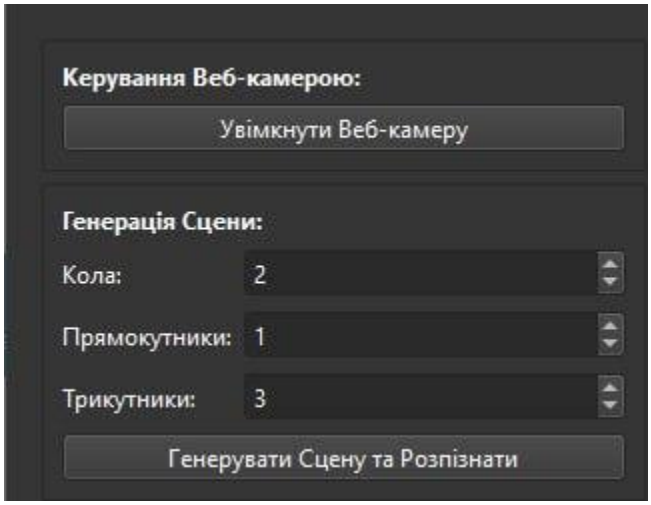


Рис. 3.13 – Панель керування режимами вебкамери та конфігурації генерації геометричних об’єктів

Тестування в режимі розпізнавання з вебкамери

Перевірка працездатності програмного застосунку в режимі реального часу здійснювалася із використанням вбудованої або зовнішньої вебкамери. Такий підхід дозволив оцінити ефективність і стабільність алгоритму розпізнавання геометричних фігур в умовах, максимально наближених до реальної експлуатації.

У процесі тестування було досліджено вплив різних факторів зовнішнього середовища на якість роботи системи, зокрема:

- ***умови освітлення:*** проведено експерименти при денному, штучному, зниженому та комбінованому освітленні. Це дало змогу оцінити адаптивність алгоритму до змін яскравості та контрасту зображення;

– ***характер фону:*** тестування здійснювалося на однотонних, текстурованих та складних фонах із наявністю сторонніх об’єктів. Це дозволило перевірити здатність системи відфільтровувати зайві елементи та коректно ідентифікувати геометричні фігури;

– ***розміри та просторове розташування об’єктів:*** використовувалися фігури різних масштабів і орієнтацій, що дало можливість оцінити ефективність алгоритму при зміні масштабу та положення об’єктів у кадрі [32];

– ***динаміка сцени:*** моделювалися ситуації з рухом об’єктів у полі зору камери, що дозволило визначити швидкодію системи та її здатність оперативно оновлювати результати розпізнавання.

Отримані результати показали, що система забезпечує швидку обробку відеопотоку та стабільне відображення результатів розпізнавання у вигляді контурів, назв фігур і їх основних характеристик. Зокрема, для кожного об’єкта визначаються такі параметри, як площа, периметр, координати центру та обмежувальна рамка (bounding box).

Водночас встановлено, що за умов значних змін освітлення точність розпізнавання може знижуватися, проте загалом алгоритм демонструє достатню стійкість до типових впливів зовнішнього середовища. Це свідчить про можливість його ефективного застосування у практичних задачах комп’ютерного зору. На рисунку 3.14 наведено приклад роботи системи в даному режимі, де коректно ідентифіковано квадрат, прямокутник, трикутник та коло з відповідними обчисленими параметрами.

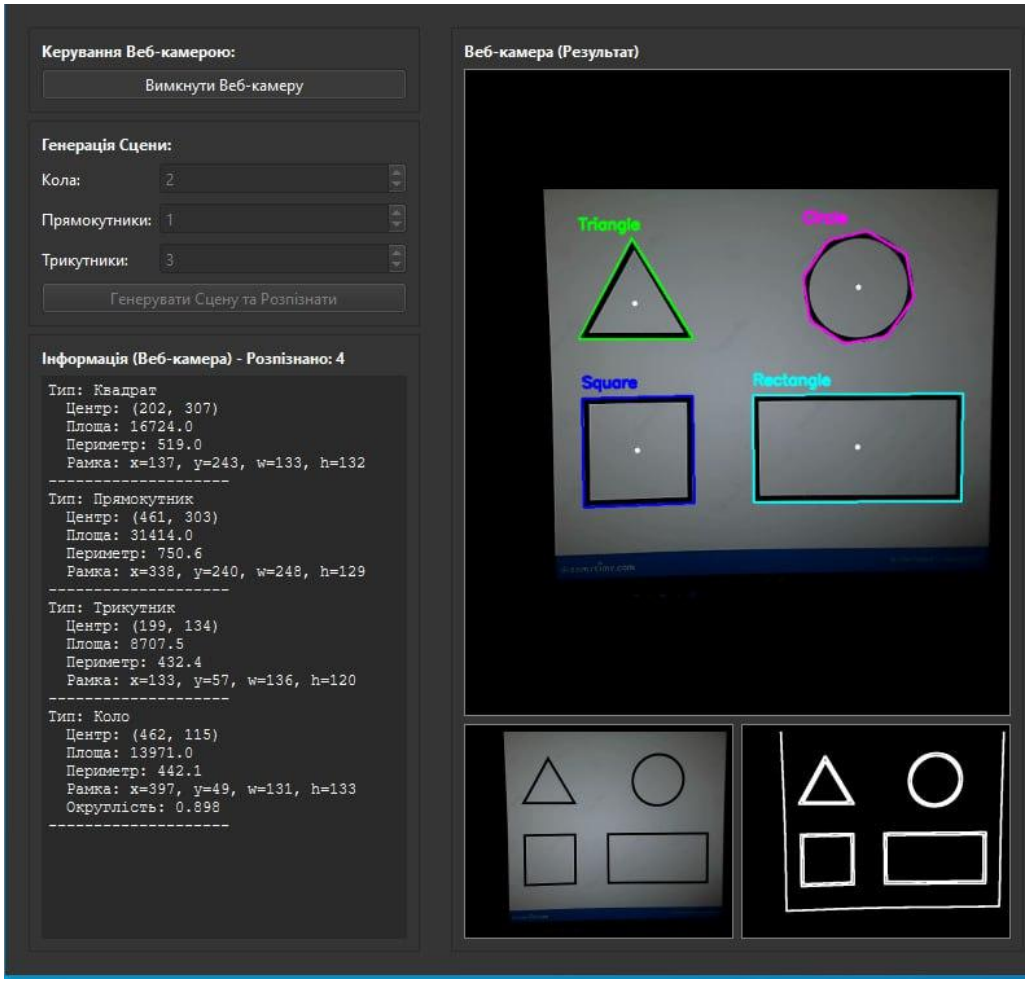


Рис. 3.14 – Результати розпізнавання геометричних фігур у режимі реального часу з вебкамери із відображенням їх параметрів

Другий підхід до тестування був спрямований на кількісну оцінку точності алгоритму розпізнавання шляхом використання синтетично сформованих сцен. Для реалізації цього підходу застосовувався спеціалізований скрипт scene_generator.py, який генерує тестові зображення із заздалегідь визначеними параметрами: кількістю геометричних фігур (кола, прямокутники, трикутники), їх розмірами та координатами розміщення на площині. Сукупність цих параметрів формує еталонне представлення (ground truth), що використовується для подальшого порівняння результатів.

Після формування сцени до зображення застосовувався алгоритм розпізнавання (shape_detector.py), який визначає:

– тип кожного об’єкта;

- координати центру фігури;
- площу, периметр та габаритні характеристики (обмежувальну рамку);
- додаткові геометричні показники, зокрема коефіцієнт округлості.

Отримані результати зіставлялися з еталонними даними, що дозволило об’єктивно оцінити точність алгоритму як у частині класифікації фігур, так і у визначенні їх параметрів. Такий підхід забезпечує проведення тестування у контрольованому середовищі, позбавленому впливу шумів, змін освітлення та інших факторів, характерних для реального відеопотоку [32].

Застосування даного методу дозволило не лише перевірити коректність функціонування системи, а й виявити її потенційні обмеження. Зокрема, було встановлено, що складнощі можуть виникати у випадках часткового перекриття об’єктів або при розпізнаванні фігур зі схожими геометричними характеристиками, наприклад квадратів і прямокутників. Таким чином, даний режим став основою для проведення кількісного аналізу ефективності алгоритму в умовах моделювання, що є важливою складовою комплексного тестування системи.

На рисунку 3.15 наведено приклад роботи застосунку в режимі розпізнавання на згенерованій сцені. У правій верхній частині інтерфейсу відображено результати обробки з автоматичним виділенням контурів, підписом типів фігур (коло, трикутник, прямокутник) та визначенням їх параметрів. У нижній частині представлено еталонне зображення, сформоване генератором сцен. Ліва панель містить текстову інформацію з аналітичними характеристиками кожного об’єкта, включаючи координати центру, площу, периметр, параметри обмежувальної рамки та коефіцієнт округлості. Така структура інтерфейсу забезпечує наочне порівняння результатів розпізнавання з еталонними даними, що є основою для об’єктивної оцінки ефективності алгоритму.

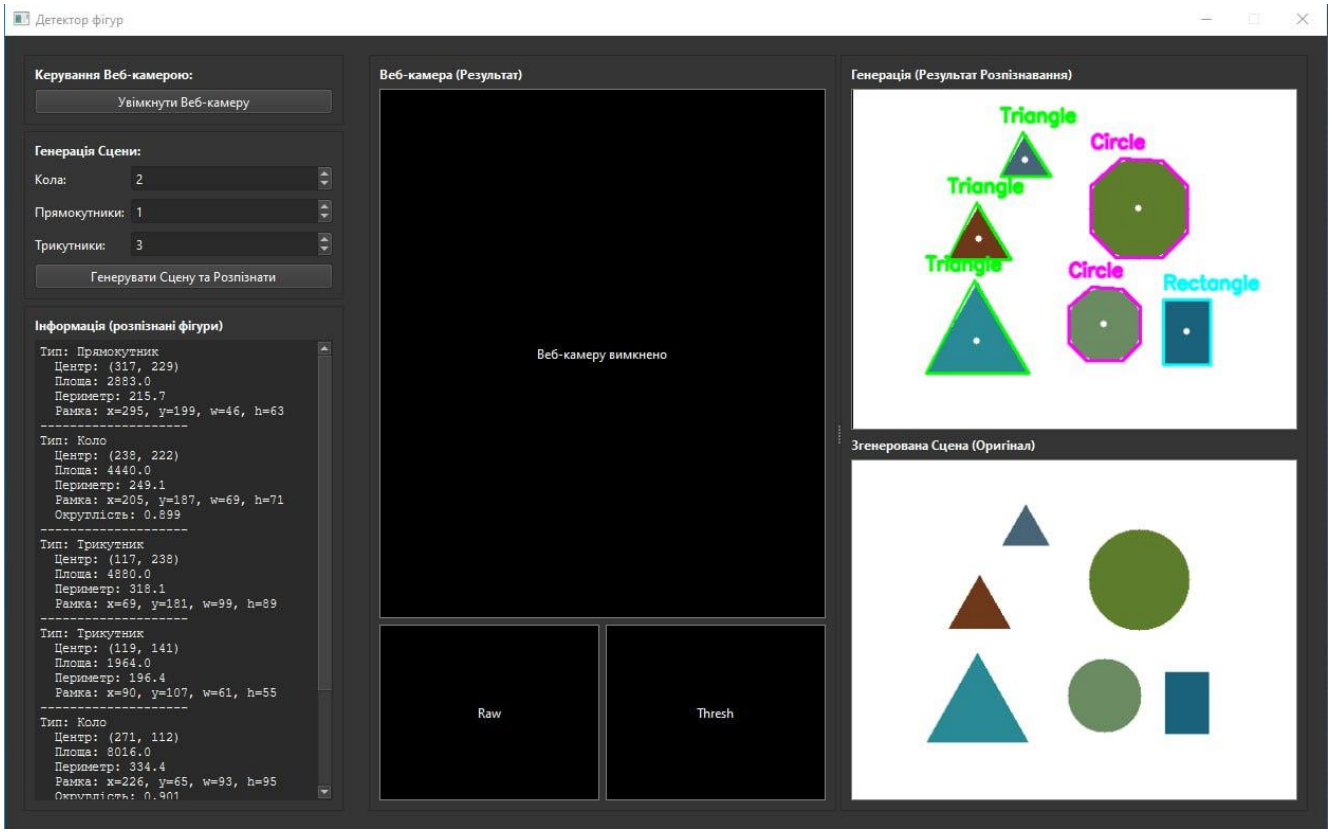


Рис. 3.15 – Відображення процесу розпізнавання геометричних об’єктів на синтетично сформованій сцені з перевіркою відповідності еталонним даним

Нижче на відео продемонстровано роботу розробленого застосунку в обох режимах: розпізнавання геометричних фігур у реальному часі за допомогою вебкамери та на основі згенерованої сцени. У режимі вебкамери видно, як система автоматично ідентифікує фігури на паперовому носії, визначає їхній тип, контур і параметри. У режимі генерації сцени — показано створення випадкової конфігурації об’єктів та подальше успішне розпізнавання з порівнянням еталонних даних [32].

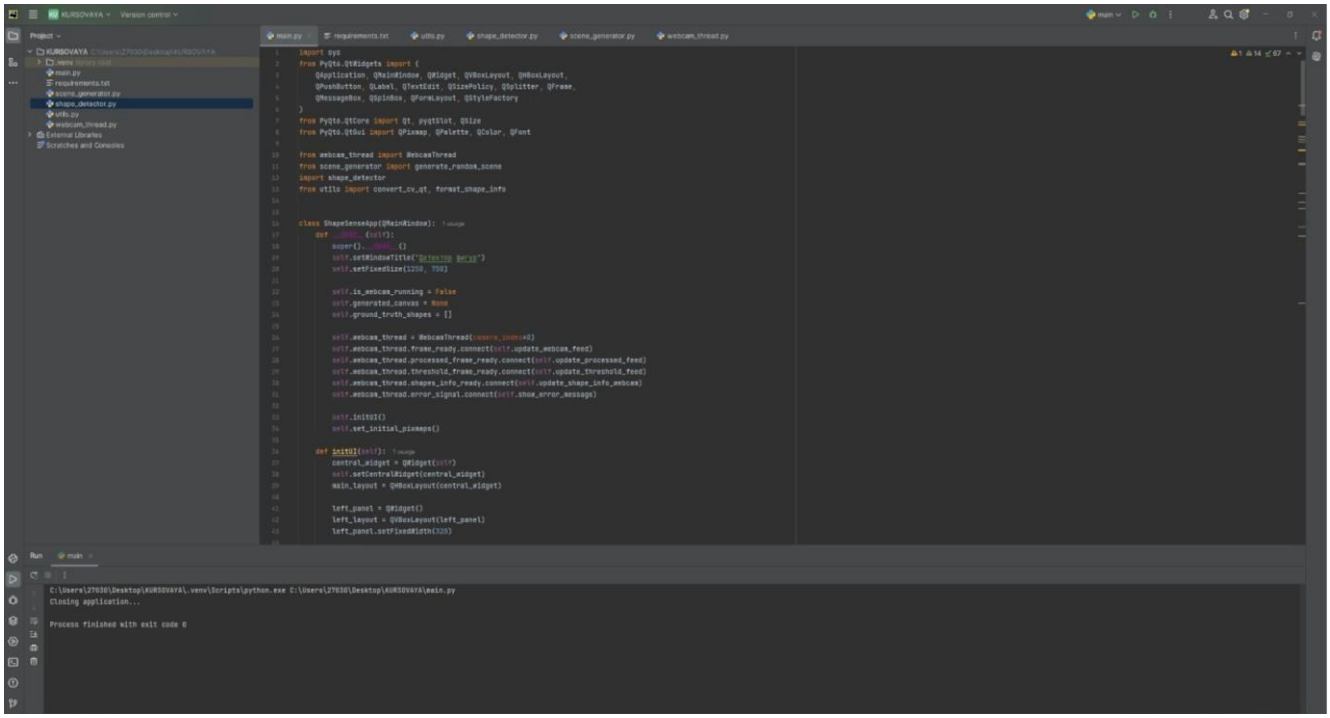


Рис. 3.16 – Програмна реалізація інтерфейсу користувача та логіки застосунку на мові Python з використанням PyQt6

3.4 Оцінювання ефективності роботи системи та напрями подальшого розвитку

У межах даного підрозділу здійснено комплексне оцінювання ефективності розробленого програмного застосунку для розпізнавання геометричних фігур, а також визначено основні напрями його подальшого вдосконалення. Аналіз ефективності проводився на основі результатів тестування у двох режимах: у реальному часі з використанням вебкамери та у контрольованому середовищі на синтетично згенерованих сценах.

Оцінювання роботи системи здійснювалося за такими основними критеріями: точність розпізнавання об’єктів, швидкість обробки зображень, стійкість до змін зовнішнього середовища, а також зручність використання інтерфейсу.

Першим важливим показником є точність розпізнавання геометричних фігур. У режимі роботи із згенерованими сценами система продемонструвала високий рівень точності, оскільки відсутність шумів та сторонніх факторів дозволяє

алгоритму працювати у контрольованих умовах. У більшості випадків алгоритм коректно визначав типи фігур (коло, трикутник, прямокутник), а також обчислював їхні параметри, такі як площа, периметр, координати центру та габаритні характеристики. Водночас було виявлено певні труднощі при класифікації об’єктів зі схожими властивостями, зокрема квадратів і прямокутників, що зумовлено особливостями аналізу контурів.

У режимі роботи з вебкамерою точність розпізнавання дещо знижувалася через вплив зовнішніх факторів. Зокрема, значний вплив мають умови освітлення, наявність шумів у зображенні, складний фон або часткове перекриття об’єктів. Проте навіть у таких умовах система демонструвала задовільні результати, що свідчить про достатній рівень адаптивності алгоритму.

Другим важливим критерієм є швидкість обробки даних. Проведене тестування показало, що застосунок здатний обробляти відеопотік у режимі реального часу без суттєвих затримок. Це досягається за рахунок використання ефективних алгоритмів обробки зображень бібліотеки OpenCV, а також оптимальної організації потоків виконання (зокрема використання окремого потоку для роботи з вебкамерою). Швидкодія системи дозволяє використовувати її для інтерактивних задач, де важливо отримувати результат миттєво.

Наступним критерієм є стійкість до впливу зовнішніх факторів. Як показали результати тестування, алгоритм зберігає працездатність при помірних змінах освітлення та наявності незначних шумів. Однак при різких змінах умов (сильне затемнення або пересвічення) якість розпізнавання може знижуватися. Це свідчить про необхідність подальшого вдосконалення методів попередньої обробки зображення.

Окрему увагу слід приділити зручності користувацького інтерфейсу. Інтерфейс застосунку реалізовано з урахуванням принципів логічної структуризації та інтуїтивності. Поділ на функціональні зони (керування, візуалізація, аналітика) забезпечує зручний доступ до основних можливостей системи. Наявність декількох режимів відображення (Raw, Thresh, результат) дозволяє користувачу глибше аналізувати процес обробки зображення.

Загалом результати оцінювання свідчать про те, що розроблена система є ефективною для задач розпізнавання базових геометричних фігур і може бути використана як навчальний або демонстраційний інструмент у сфері комп’ютерного зору.

Разом із тим, проведений аналіз дозволив визначити низку напрямів подальшого розвитку системи.

Перш за все, доцільним є підвищення точності розпізнавання за рахунок використання більш складних алгоритмів. Зокрема, можна інтегрувати методи машинного навчання або нейронні мережі, які здатні краще працювати в умовах шумів та складних сцен.

Другим напрямом є удосконалення попередньої обробки зображень. Це може включати застосування адаптивних методів бінаризації, фільтрації шумів, нормалізації освітлення, що дозволить підвищити стабільність роботи алгоритму в реальних умовах.

Третім напрямом розвитку є розширення функціональності застосунку. Зокрема, можна додати підтримку розпізнавання складніших геометричних об’єктів, багатокутників або комбінованих фігур, а також реалізувати можливість роботи з відеофайлами або зображеннями з файлової системи.

Ще одним перспективним напрямом є оптимізація продуктивності. Використання апаратного прискорення (наприклад, GPU) або паралельних обчислень може значно підвищити швидкість обробки даних.

Крім того, доцільним є розширення інтерфейсу користувача, зокрема додавання графіків, статистики або можливості збереження результатів аналізу. Це зробить систему більш зручною для дослідницьких задач.

Висновок до розділу 3

У третьому розділі було реалізовано програмний застосунок для розпізнавання геометричних фігур на основі методів комп’ютерного зору. Розглянуто архітектуру системи, структуру основних програмних модулів, а також особливості реалізації користувацького інтерфейсу із використанням бібліотеки PyQt6. Описано взаємодію між компонентами застосунку, включаючи обробку відеопотоку, генерацію синтетичних сцен та виведення аналітичної інформації про розпізнані об’єкти.

У процесі дослідження було проведено тестування системи у двох режимах: у реальному часі з використанням вебкамери та у контрольованому середовищі на згенерованих сценах. Отримані результати показали, що розроблений алгоритм забезпечує коректне визначення основних геометричних фігур і їх параметрів, демонструючи достатню швидкодію та стійкість до типових змін зовнішнього середовища. Водночас виявлено окремі обмеження, пов’язані з впливом освітлення, шумів і складності сцени.

Проведене оцінювання ефективності підтвердило доцільність обраного підходу та практичну цінність розробленого застосунку. Запропонована система може бути використана як навчальний інструмент або основа для подальших досліджень у сфері комп’ютерного зору. Визначені напрями розвитку відкривають можливості для підвищення точності, розширення функціональності та адаптації системи до більш складних прикладних задач.

ВИСНОВКИ

У даній кваліфікаційній роботі було досліджено теоретичні та практичні аспекти розпізнавання зображень у системах комп'ютерного зору з подальшою розробкою програмного застосунку для визначення геометричних фігур. Актуальність теми зумовлена широким застосуванням технологій комп'ютерного зору у сучасних інформаційних системах, де автоматизований аналіз візуальних даних відіграє важливу роль у різних сферах діяльності.

У першому розділі було розглянуто основні теоретичні підходи до обробки зображень, проаналізовано розвиток технологій комп'ютерного зору та сфери їх застосування. Окрему увагу приділено алгоритмам виявлення об'єктів, зокрема методам виділення контурів, бінаризації та аналізу геометричних характеристик. Також проведено аналіз можливостей сучасних бібліотек, таких як OpenCV та PyQt, які забезпечують ефективну реалізацію програмних систем у даній предметній області.

У другому розділі здійснено проєктування програмної системи розпізнавання геометричних об'єктів. Було розроблено функціональну структуру застосунку, визначено основні модулі та описано їх взаємодію. Особливу увагу приділено алгоритмічній моделі визначення геометричних фігур, а також організації потокової обробки відеоданих, що забезпечує роботу системи в режимі реального часу.

У третьому розділі реалізовано програмний застосунок та проведено його тестування в різних умовах. Розроблено зручний користувацький інтерфейс, який дозволяє працювати як із відеопотоком з вебкамери, так і з синтетично згенерованими сценами. Результати експериментального дослідження показали, що система забезпечує коректне розпізнавання базових геометричних фігур і їх параметрів, демонструючи достатню швидкодію та стійкість до типових змін зовнішнього середовища.

Проведене оцінювання ефективності підтвердило доцільність використання обраних методів і технологій. Водночас було виявлено окремі обмеження,

пов’язані з впливом умов освітлення, шумів та складності сцени на точність розпізнавання. Це визначає необхідність подальшого вдосконалення алгоритмів та методів попередньої обробки зображень.

У результаті виконаної роботи досягнуто поставленої мети — розроблено програмний застосунок для розпізнавання геометричних фігур на основі методів комп’ютерного зору. Запропоноване рішення може бути використане у навчальних цілях, а також як основа для подальших досліджень і розвитку систем комп’ютерного зору з розширеним функціоналом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bradski G., Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library [Електронний ресурс]. – O'Reilly Media, 2008. – Режим доступу: https://www.oreilly.com/library/view/learning-opencv/9780596516130/ (дата звернення: 29.03.2026).
2. OpenCV Organization. OpenCV Documentation [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/ (дата звернення: 29.03.2026).
3. Szeliski R. Computer Vision: Algorithms and Applications [Електронний ресурс]. – Springer, 2010. – Режим доступу: https://szeliski.org/Book/ (дата звернення: 29.03.2026).
4. Gonzalez R. C., Woods R. E. Digital Image Processing [Електронний ресурс]. – Pearson, 2018. – Режим доступу: https://www.pearson.com/ (дата звернення: 29.03.2026).
5. PyQt Documentation. PyQt5 Reference Guide [Електронний ресурс]. – Режим доступу: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата звернення: 29.03.2026).
6. Qt Company. Qt Documentation [Електронний ресурс]. – Режим доступу: https://doc.qt.io/ (дата звернення: 29.03.2026).
7. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8, No. 6. – P. 679–698.
8. Forsyth D. A., Ponce J. Computer Vision: A Modern Approach [Електронний ресурс]. – Pearson, 2012. – Режим доступу: https://www.pearson.com/ (дата звернення: 29.03.2026).
9. Prince S. J. D. Computer Vision: Models, Learning, and Inference [Електронний ресурс]. – Cambridge University Press, 2012. – Режим д https://www.cambridge.org/ (дата звернення: 29.03.2026).
10. Goodfellow I., Bengio Y., Courville A. Deep Learning [Електронний ресурс]. – MIT Press, 2016. – Режим доступу: https://www.deeplearningbook.org/ (дата звернення: 29.03.2026).

11. Python Software Foundation. Python Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/ (дата звернення: 29.03.2026).

12. Intel Corporation. OpenCV Library Overview [Електронний ресурс]. – Режим доступу: https://opencv.org/ (дата звернення: 29.03.2026).

13. Davies E. R. Computer Vision: Principles, Algorithms, Applications, Learning [Електронний ресурс]. – Academic Press, 2012. – Режим доступу: https://www.sciencedirect.com/ (дата звернення: 29.03.2026).

14. Hartley R., Zisserman A. *Multiple View Geometry in Computer Vision* [Електронний ресурс]. – Cambridge University Press, 2004. – Режим доступу: https://www.cambridge.org/ (дата звернення: 29.03.2026).

15. Bishop C. M. *Pattern Recognition and Machine Learning* [Електронний ресурс]. – Springer, 2006. – Режим доступу: https://link.springer.com/ (дата звернення: 29.03.2026).

16. Kaehler A., Bradski G. *OpenCV 3 Computer Vision Application Programming Cookbook* [Електронний ресурс]. – Packt Publishing, 2016. – Режим доступу: https://www.packtpub.com/ (дата звернення: 29.03.2026).

17. Rosebrock A. *Practical Python and OpenCV* [Електронний ресурс]. – PyImageSearch, 2016. – Режим доступу: https://pyimagesearch.com/ (дата звернення: 29.03.2026).

18. OpenCV. Open Source Computer Vision Library [Електронний ресурс]. – Режим доступу: https://opencv.org/ (дата звернення: 15.03.2026).

19. Bradski G., Kaehler A. Learning OpenCV 4: Computer Vision with Python. – Sebastopol: O'Reilly Media, 2020. – 754 p.

20. PyQt6 Documentation [Електронний ресурс]. – Режим доступу: https://www.riverbankcomputing.com/static/Docs/PyQt6/ (дата звер 15.03.2026).

21. Qt Company. Qt Documentation [Електронний ресурс]. – Режим доступу: https://doc.qt.io/ (дата звернення: 15.03.2026).

22. Szeliski R. Computer Vision: Algorithms and Applications. – 2nd ed. – Cham: Springer, 2022. – 832 p.

23. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – New York: Pearson, 2018. – 1168 p.

24. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8, No. 6. – P. 679–698.

25. PyImageSearch. OpenCV Tutorials [Електронний ресурс]. – Режим доступу: https://pyimagesearch.com/ (дата звернення: 16.03.2026).

26. Python Software Foundation. Python Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/ (дата звернення: 15.03.2026).

27. NumPy Developers. NumPy Documentation [Електронний ресурс]. – Режим доступу: https://numpy.org/doc/ (дата звернення: 15.03.2026).

28. Intel Corporation. OpenCV Canny Edge Detection [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (дата звернення: 16.03.2026).

29. OpenCV Documentation. Contours in OpenCV [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html (дата звернення: 16.03.2026).

30. OpenCV Documentation. Shape Detection [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/4.x/ (дата звернення: 16.03.2026).

31. Hartley R., Zisserman A. Multiple View Geometry in Computer Vision. – 2nd ed. – Cambridge: Cambridge University Press, 2004. – 655 p.

32. Bishop C. M. Pattern Recognition and Machine Learning. – New York: Springer, 2006. – 738 p.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА «ПРОГРАМНИЙ ЗАСТОСУНОК РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ НА ОСНОВІ МЕТОДІВ КОМП'ЮТЕРНОГО ЗОРУ»

Виконав студент: групи ІСД-42 Віолетта ФОМЕНКО

Керівник роботи: PhD доцент Валентина ДАНИЛЬЧЕНКО

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

2

Мета роботи:

Створення програмної системи для автоматичного розпізнавання зображень, яка здійснює обробку відеопотоку з вебкамери або згенерованих сцен, визначає типи геометричних фігур та відображає інформацію про них у зручному інтерфейсі користувача.

Об'єкт дослідження:

Процеси автоматизованої обробки та аналізу графічних даних із застосуванням технологій комп'ютерного зору.

Предмет дослідження:

Алгоритмічні методи та програмні інструменти розпізнавання геометричних об'єктів у режимі реального часу з використанням бібліотеки OpenCV та графічного інтерфейсу PyQt.

НАУКОВІ ЗАВДАННЯ ТА МЕТОДИ ДОСЛІДЖЕННЯ 3

Наукові завдання:

- Аналіз сучасних підходів до обробки та розпізнавання зображень у системах комп'ютерного зору
- Аналіз алгоритмів виявлення об'єктів і вибір оптимальних для реалізації
- Дослідження можливостей Python, OpenCV та PyQt для систем КЗ
- Розробка алгоритму визначення геометричних фігур за контурами
- Проєктування архітектури програмного застосунку
- Реалізація застосунку з обробкою відеопотоку в реальному часі
- Тестування системи та оцінка ефективності роботи

Методи дослідження:

- Дослідження сучасних підходів до реалізації систем КЗ з Python та OpenCV
- Розроблення алгоритму виділення контурів та класифікації геометричних об'єктів
- Алгоритм Кенні (Canny) для виявлення меж об'єктів
- Апроксимація Дугласа-Пекера для спрощення контурів
- Реалізація PyQt-інтерфейсу та генератора тестових сцен
- Експериментальне тестування на відеопотоці та синтетичних сценах

АНАЛІЗ ТЕХНОЛОГІЙ ТА БІБЛІОТЕК ДЛЯ СИСТЕМ КОМП'ЮТЕРНОГО ЗОРУ 4

Таблиця 1 – Порівняльна характеристика бібліотек OpenCV та PyQt

Характеристика	OpenCV	PyQt
Призначення	Обробка зображень та відео	Створення графічного інтерфейсу
Основні функції	Фільтрація, контури, Canny, threshold	Вікна, кнопки, GUI-компоненти
Тип задач	Комп'ютерний зір, аналіз зображень	Взаємодія з користувачем
Переваги	Висока швидкість, готові алгоритми CV	Зручний UI, інтерактивність
Роль у роботі	Розпізнавання геометричних фігур	Відображення результатів

Рисунок 1 – Загальна схема роботи системи комп'ютерного зору

АЛГОРИТМ ОБРОБКИ ЗОБРАЖЕННЯ У СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ 5

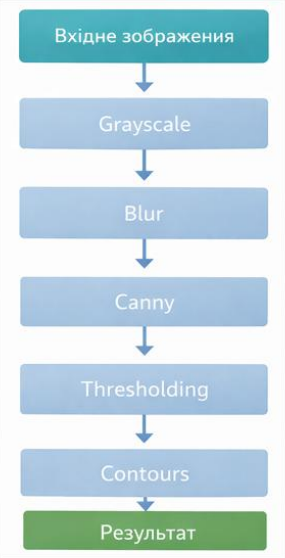


Рисунок 2 – Загальна схема алгоритму обробки зображення

ФУНКЦІОНАЛЬНА СТРУКТУРА ТА МОДУЛЬНА АРХІТЕКТУРА ЗАСТОСУНКУ 6

Рисунок 3 – Функціональна структура програмного застосунку

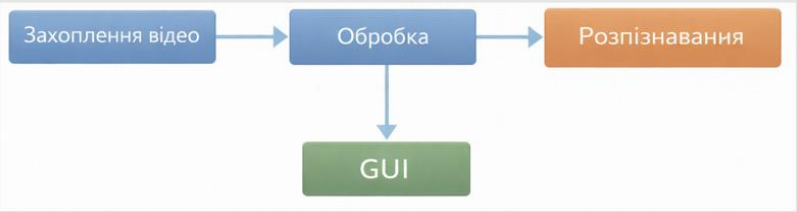


Рисунок 4 – Модульна архітектура системи

АЛГОРИТМ ВИЗНАЧЕННЯ ГЕОМЕТРИЧНИХ ФІГУР 7

Рисунок 5 – Схематичне представлення алгоритму визначення геометричних фігур

ОПИС ПРОГРАМНИХ МОДУЛІВ ТА АРХІТЕКТУРА ПРОЄКТУ 8

Таблиця 2 – Опис основних модулів системи розпізнавання зображень

Модуль	Файл	Основні функції
Головний застосунок	main.py	Ініціалізація GUI, координація модулів, ShapeSenseApp
Генерація сцен	scene_generator.py	Формування тестових зображень з фігурами (ground truth)
Розпізнавання	shape_detector.py	detect_shapes(): Canny, findContours, approxPolyDP, класифікація
Допоміжний модуль	utils.py	convert_cv_qt(), format_shape_info() — відображення результатів
Потік вебкамери	webcam_thread.py	WebcamThread (QThread): асинхронне захоплення відеопотоку

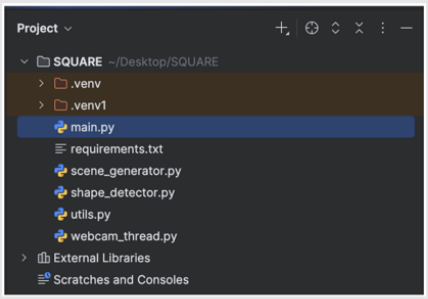


Рисунок 6 – Архітектурна структура програмного проєкту системи розпізнавання

ВИМОГИ ДО ПРОГРАМНОГО ЗАСТОСУНКУ 9

Таблиця 3 – Функціональні вимоги до системи розпізнавання зображень

№	Вимога	Опис
1	Обробка відеопотоку	Захоплення кадрів з вебкамери в режимі реального часу (QThread)
2	Генерація тестових сцен	Синтетичні зображення з колами, прямокутниками, трикутниками
3	Розпізнавання фігур	Визначення типу: трикутник (3), квадрат/прямокутник (4), коло (>5 вершин)
4	Обчислення характеристик	Площа, периметр, центр мас, bounding box, коефіцієнт округлості
5	Відображення результатів	GUI: оброблений кадр + raw + thresh + текстова інформація

Таблиця 4 – Нефункціональні вимоги до системи розпізнавання зображень

№	Вимога	Опис
1	Швидкодія	Обробка відеопотоку у реальному часі без суттєвих затримок
2	Модульність	Чіткий поділ на незалежні модулі (main, detector, generator, utils, thread)
3	Стійкість	Збереження працездатності при змінах освітлення та шумах
4	Зручність GUI	Логічна структура інтерфейсу: керування, візуалізація, аналітика

ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ: РЕЖИМ ВЕБКАМЕРИ 10

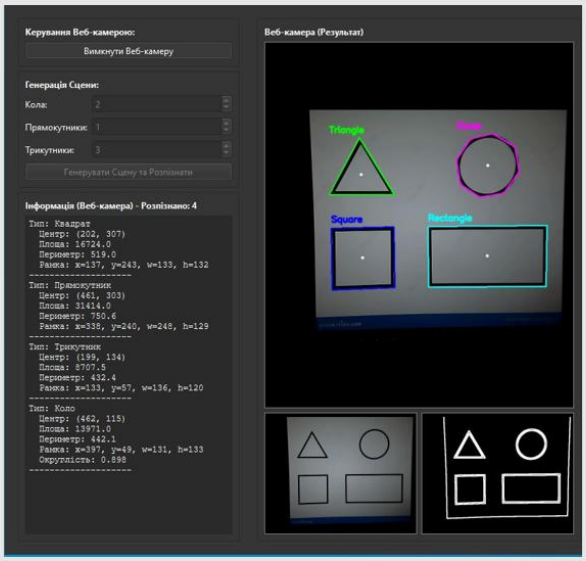


Рисунок 7 – Результати розпізнавання геометричних фігур у режимі реального часу з вебкамери

ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ: РЕЖИМ ЗГЕНЕРОВАНОЇ СЦЕНИ

11

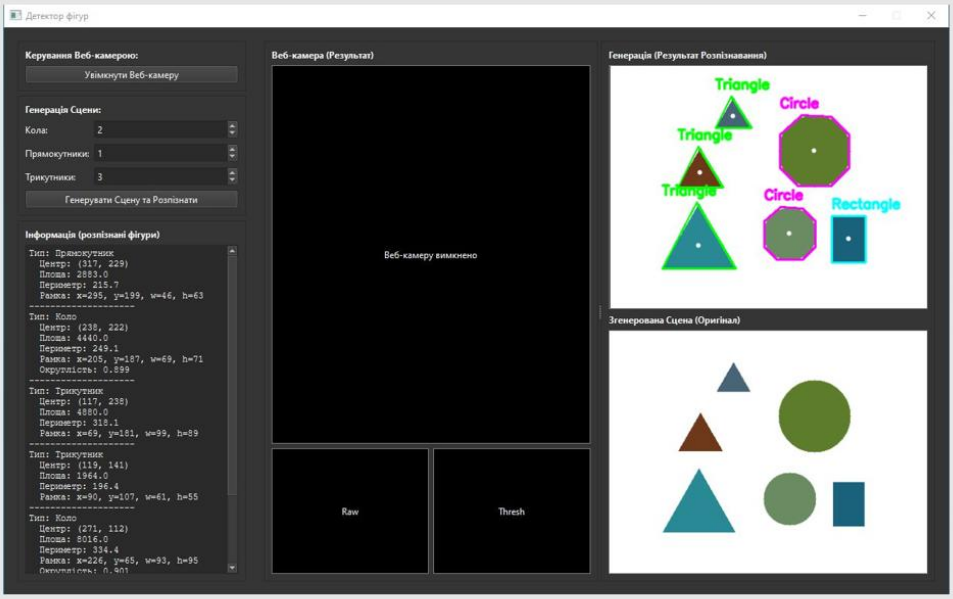


Рисунок 8 – Відображення процесу розпізнавання на синтетично сформованій сцені з порівнянням еталонних даних

ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

12

Режим вебкамери

- Коректне розпізнавання трикутників, кіл, квадратів та прямокутників
- Стабільна робота при денному та штучному освітленні
- Обробка у реальному часі без суттєвих затримок
- Виведення параметрів: площа, периметр, центр, bounding box
- Зниження точності при різких змінах освітлення або складному фоні

Режим генерації сцен

- Висока точність завдяки відсутності шумів та контрольованому середовищу
- Порівняння результатів із еталонними даними (ground truth)
- Труднощі при класифікації квадратів та прямокутників зі схожими пропорціями
- Ефективний інструмент для навчальних та демонстраційних цілей
- Можливість задавати кількість кіл, прямокутників та трикутників

Напрями подальшого розвитку: інтеграція нейронних мереж, адаптивна бінаризація, GPU-прискорення, підтримка відеофайлів.

ВИСНОВКИ

13

1. Розглянуто теоретичні основи комп'ютерного зору та проаналізовано алгоритми виявлення об'єктів — виділення меж (Canny), бінаризацію, пошук і апроксимацію контурів
2. Проаналізовано бібліотеки OpenCV та PyQt і обґрунтовано їх поєднання: OpenCV — для алгоритмічної обробки, PyQt — для графічного інтерфейсу
3. Розроблено алгоритм класифікації геометричних фігур за кількістю вершин та коефіцієнтом округлості (трикутник, квадрат, прямокутник, коло)
4. Спроєктовано модульну архітектуру застосунку (main.py, shape_detector.py, scene_generator.py, webcam_thread.py, utils.py)
5. Реалізовано два режими роботи: розпізнавання у реальному часі з вебкамери та на синтетично згенерованих сценах
6. Тестування підтвердило достатню точність і швидкодію системи; виявлено обмеження при складних умовах освітлення
7. Розроблений застосунок може бути використаний у навчальних цілях та як основа для подальших досліджень у сфері КЗ

АПРОБАЦІЯ:

1. III Всеукраїнська науково-технічна конференція, ДУІКТ, «Застосування машинного навчання у системах відеоаналітики та розпізнавання облич для забезпечення безпеки» на тему: «Застосування машинного навчання у системах відеоаналітики та розпізнавання облич для забезпечення безпеки».

2. VII Міжнародній науково-технічній конференції «Сучасний стан та перспективи розвитку IoT» на тему: «Застосування методів комп’ютерного зору для автоматизованого аналізу візуальних даних».

ДЯКУЮ ЗА УВАГУ!