МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана ШОВКОПЛЯС
(підпис)
6 грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»
освітньо-професійної програми «Інформатика»
на тему: Інформаційна технологія детектування об'єкту інтересу на місцевості за зразками його зображення
здобувача групи ІН.м - 34 Бублика Владислава Олеговича

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

Владислав БУБЛИК
(підпис)

Керівник
кандидат наук, доцент
В'ячеслав МОСКАЛЕНКО
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 «Комп'ютерні науки», освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-34 Бублика Владислава Олеговича

1. Тема роботи: ««Інформаційна технологія детектування об'єкту інтересу на місцевості за зразками його зображення»
затверджую наказом по СумДУ від «03» грудня 2024 року №1257-VI
2. Термін здачі здобувачем кваліфікаційної роботи *до 06 грудня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для виявлення об'єктів 3) Розробка інформаційної технології з детектування об'єкту інтересу на місцевості за зразком 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Аналіз проблеми предметної області, постановка й формування завдань дослідження	19.08-31.08	
2	Огляд технологій, що використовуються для виявлення об'єктів	01.09-14.10	
3	Розробка інформаційної технології з детектування об'єкту інтересу на місцевості за зразками його зображення	15.10-10.11	
4	Аналіз отриманих результатів	11.11-19.11	
5	Оформлення пояснювальної записки до кваліфікаційної роботи	20.11-01.12	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 60 стр., 59 рис., 1 додаток, 21 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі детектування й розпізнавання об'єктів ворожої техніки шляхом розробки відповідних методів, моделей та інформаційної технології.

Об'єкт дослідження — процес автоматизованого виявлення та ідентифікації ворожої техніки на місцевості розвідувальними БПЛА.

Мета роботи — розробка інформаційної технології детектування й розпізнавання об'єктів ворожої техніки за зразком її зображення на основі нейромереж.

Методи дослідження — алгоритми комп'ютерного зору для детектування і порівняння об'єктів на зображеннях.

Результати — розроблено інформаційну технологію, яка виявляє об'єкти ворожої техніки на місцевості, обробляє їх і розпізнає потрібний шляхом порівняння зі зразком його зображення. Проведено тестування розробки на реальних фотографіях з дронів.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ДЕТЕКТУВАННЯ, ЕКСТРАКТОР ОЗНАК, YOLO, SUPERPOINT, SUPERGLUE, PYTHON.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми роботи пов’язане зі зростаючою потребою в ефективних та високоточних технологіях для автоматизованого виявлення необхідних об’єктів у складних і динамічних умовах.

Актуальність обраної теми полягає в необхідності створення систем автоматизованого розпізнавання об’єктів на місцевості, на основі новітніх технологій. Це може бути корисно в багатьох сферах, таких як безпека, охорона важливих об’єктів, контроль інфраструктурних об’єктів, дослідження природних ресурсів, а також у військових операціях. Покращення процесу ідентифікації об'єктів дозволяє значно підвищити ефективність систем та знизити залежність від людського фактора.

Об’єкт дослідження. Процес автоматизованого виявлення та ідентифікації ворожої техніки на місцевості за допомогою безпілотних літальних апаратів.

Предмет дослідження. Методологія та алгоритми для забезпечення точності, швидкості та стійкості системи детектування об’єктів на зображенні.

Новизна. Одним з ключових викликів у цій сфері є розробка технологій, що можуть забезпечити точне та стабільне виявлення об'єктів при різних умовах, таких як зміни освітлення, різні ракурси або сторонні об'єкти. В наш час ефективним методом для вирішення цієї задачі є використання нейронних мереж, які дозволяють виявляти об’єкт, знаходити та порівнювати ключові точки на зображеннях, а потім відстежувати їх. Такі технології дозволяють обробляти великі обсяги даних в режимі реального часу, що відкриває широкі перспективи для їх впровадження в системах з різними вимогами до продуктивності та точності.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибору методу розв’язання поставленої задачі, опису

програмного забезпечення інформаційної технології, висновків, списку використаних джерел та додатків.

1. АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Сучасний стан та тенденції розвитку інтелектуальних систем пошуку об’єктів інтересу на аерозображенні

В умовах швидкого прогресу технологічного середовища інтелектуальні системи пошуку об'єктів інтересу на аерозображеннях стали невід'ємною частиною багатьох галузей життя. Для цієї задачі використовують безпілотні апарати, які є незамінними інструментами в сферах екології, охорони та військових операцій. Детектування об’єктів інтересу та їх класифікація є корисними для таких практичних завдань, як оцінка стану території, відео-аналіз систем рятувально-пошукових заходів, охорона периметру, наведення зброї [1].

Екстрактори ознак відіграють критичну роль в системах комп'ютерного зору. Саме вони дозволяють системам автоматично ідентифікувати, класифікувати та відстежувати об'єкти, використовуючи інформацію з зображень [2]. Екстрактори ознак — це алгоритми або модулі, які виділяють такі релевантні характеристики з зображення, як: кольори, текстури, градієнти, контури та інші візуальні елементи, що допомагають у розпізнаванні об'єктів [3].

Процес навчання моделей аналізу й розпізнавання об'єктів на місцевості здійснюється за допомогою апаратних прискорювачів на основі графічних обчислювальних пристроїв і нейронних процесорів і є досить ресурсномістким, що ускладнює їх використання в автономному режимі. Це визначає актуальність полішення інструментів і методів навчання, адаптованих до умов функціонування у випадку ресурсних обмежень [4].

З розвитком технологій екстрактори ознак стають більш складними та ефективними, що суттєво впливає на якість і швидкість аналізу зображень. На початку комп'ютерного зору основні методи екстракції базувалися на статистичних та геометричних характеристиках. Вони працювали добре за

певних умов, але мали обмеження щодо варіацій в освітленні, масштабі та кутах зору. Зараз мережі можуть автоматично виділяти важливі ознаки на різних рівнях абстракції, що забезпечує вищу точність виявлення об'єктів [5].

Однією з основних тенденцій у розвитку цих систем є інтеграція штучного інтелекту. Системи, що використовують глибоке навчання, стали особливо популярними через їх здатність автоматично виявляти та класифікувати об'єкти з високою точністю. Наприклад, алгоритми на основі нейронних мереж, такі як YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), демонструють вражаючу продуктивність у реальному часі, що є критично важливим для оперативних військових завдань та моніторингу природних катастроф.

Також існує зростаючий інтерес до використання методів графового навчання для покращення процесів виявлення об'єктів. Алгоритми, такі як SuperPoint-SuperGlue, використовують графові нейронні мережі для співставлення ознак між зображеннями, що забезпечує високу точність у складних умовах, де традиційні методи можуть зазнавати труднощів. Цей підхід дозволяє покращити стійкість системи до змін освітлення та ракурсів, які часто виникають при зборі аерозображень [6].

Важливою тенденцією в цій сфері є впровадження технологій зворотного зв'язку та автоматизованого навчання. Це дозволяє системам адаптуватися до нових умов та типів об'єктів, що вимагає постійного оновлення моделей і даних. Використання великих обсягів даних, що збираються під час польотів дронів, відкриває можливості для тренування моделей на реальних прикладах, що підвищує їх точність і ефективність [5].

Таким чином, сучасні інтелектуальні системи пошуку об'єктів на аерозображеннях базуються на передових методах аналізу зображень та машинного навчання, що забезпечує їх високу продуктивність та адаптивність. З огляду на постійний розвиток технологій, можна з упевненістю

стверджувати, що подальші досягнення в цій сфері принесуть нові можливості для поліпшення систем виявлення та моніторингу об'єктів.

1.2. Аналіз моделей і методів екстракції візуальних ознак

Екстракція візуальних ознак є ключовим етапом в аналізі зображень, що забезпечує системам машинного зору можливість виявлення та класифікації об'єктів. Протягом останніх десятиліть розвиток цих методів суттєво прогресує, і сьогодні існують різноманітні моделі, що варіюються від класичних детекторів ознак до сучасних глибинних нейронних мереж.

1.2.1. Класичні методи екстракції ознак

На ранніх етапах розвитку комп'ютерного зору основну увагу приділяли розробці методів, що ґрунтувалися на математичних і геометричних принципах. Ці методи, незважаючи на їхню простоту, виявилися ефективними для певних задач і відіграли важливу роль у формуванні базових концепцій екстракції ознак [7]. Відомими класичними інструментами екстракції ознак є: SIFT, SURF та HOG.

SIFT (Scale-Invariant Feature Transform) був розроблений Девідом Лоу у 1999 році і став одним із перших методів екстракції ознак, який досяг стійкості до масштабів і обертання зображень. Алгоритм SIFT складається з декількох кроків:

- Алгоритм використовує гауссові похідні для побудови масштабної піраміди зображення та виділяє екстремальні значення в різних масштабах.
- Для кожної ключової точки визначається орієнтація, що робить цей метод стійким до обертання зображення.
- Будується вектор ознак (128 вимірів), що кодує інформацію про орієнтацію, масштаби та градієнти в околі точки [8].

Перевагами SIFT є висока точність та стійкість до змін масштабу, обертання, та часткових деформацій. Недоліком вважається повільна робота через велику кількість обчислень, що робить його непридатним для реального часу.

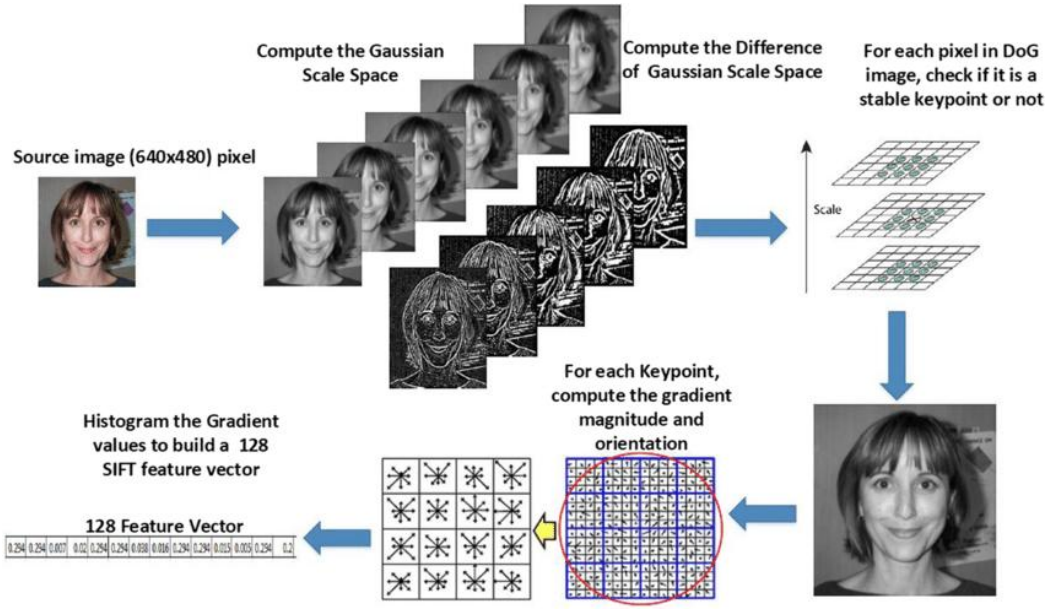


Рисунок 1.1 – Алгоритм роботи SIFT

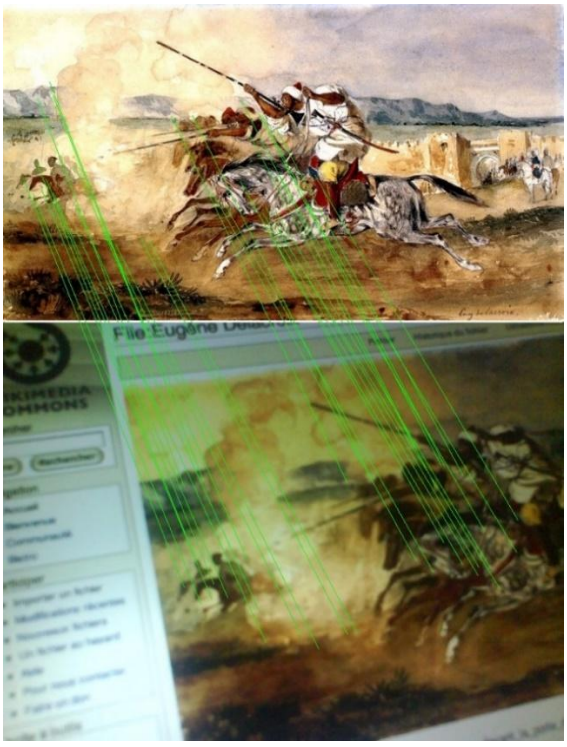


Рисунок 1.2 – Відповідність ключових точок, знайдених за допомогою SIFT

SURF (Speeded Up Robust Features), розроблений у 2006 році і опублікований Гербертом Беєм, Тінне Туйтелаарс та Люком Ван Ґулом, — це виявляч та описувач локальних ознак. Він став удосконаленим і швидшим аналогом SIFT. Основна мета цього методу — прискорення обчислень без значної втрати точності. SURF використовує гессіан-матриці для пошуку точок інтересу та апроксимацію гауссових фільтрів з використанням інтегральних зображень для швидшого обчислення [9].

Зображення перетворюється на координати методом піраміди з декількома роздільностями, для копіювання основної фотографії за допомогою піраміди гауссового або лапласового вигляду і отримання зображення такого ж розміру, але меншої роздільної здатності. Так досягається ефект розмиття первинного зображення, який називається простором масштабів, і який забезпечує масштабну інваріантність особливих точок [10].

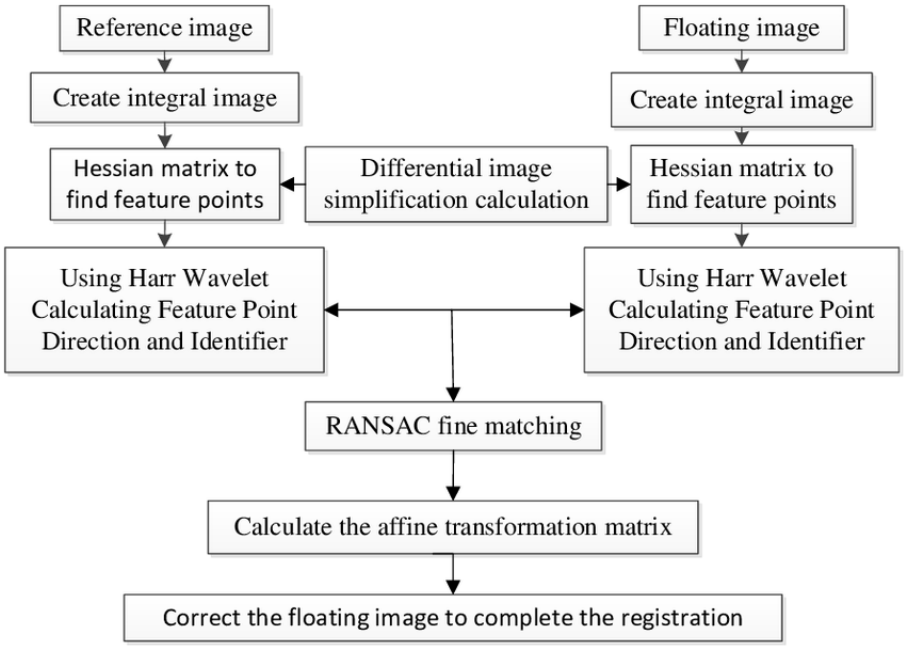


Рисунок 1.3 – Алгоритм роботи SURF

Перевагою SURF є те, що він добре працює в задачах, де вимір часу критичний. Недоліком є менша стійкість до змін освітлення.

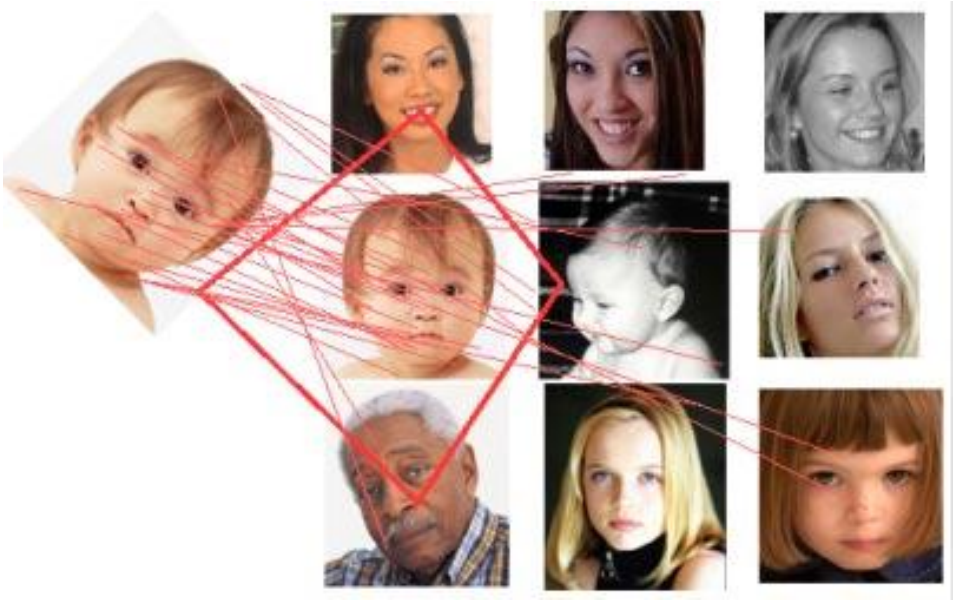


Рисунок 1.4 – Приклад роботи SURF

HOG (Histogram Of Oriented Gradients), представлений Навнітом Далалом та Біллом Тріггсом у 2005 році, також є одним із класичних методів для виявлення об'єктів. Метод ґрунтується на побудові гістограм напрямків градієнтів для кожної частини зображення, що дозволяє ефективно виділяти контури об'єктів.

Зображення розбивається на невеликі області, які називаються комірками, й для пікселів, у кожній з них, складається гістограма напрямків градієнта. Даний дескриптор — це з'єднання цих гістограм. Для підвищення точності локальні гістограми можуть унормовуватись за контрастом, обчислюючи яскравість в більшій частині зображення й використовуючи потім це значення міри для унормовування всіх клітинок у блоці. Ця нормалізація забезпечує кращу інваріантність по відношенню до змін освітлення та затінення.

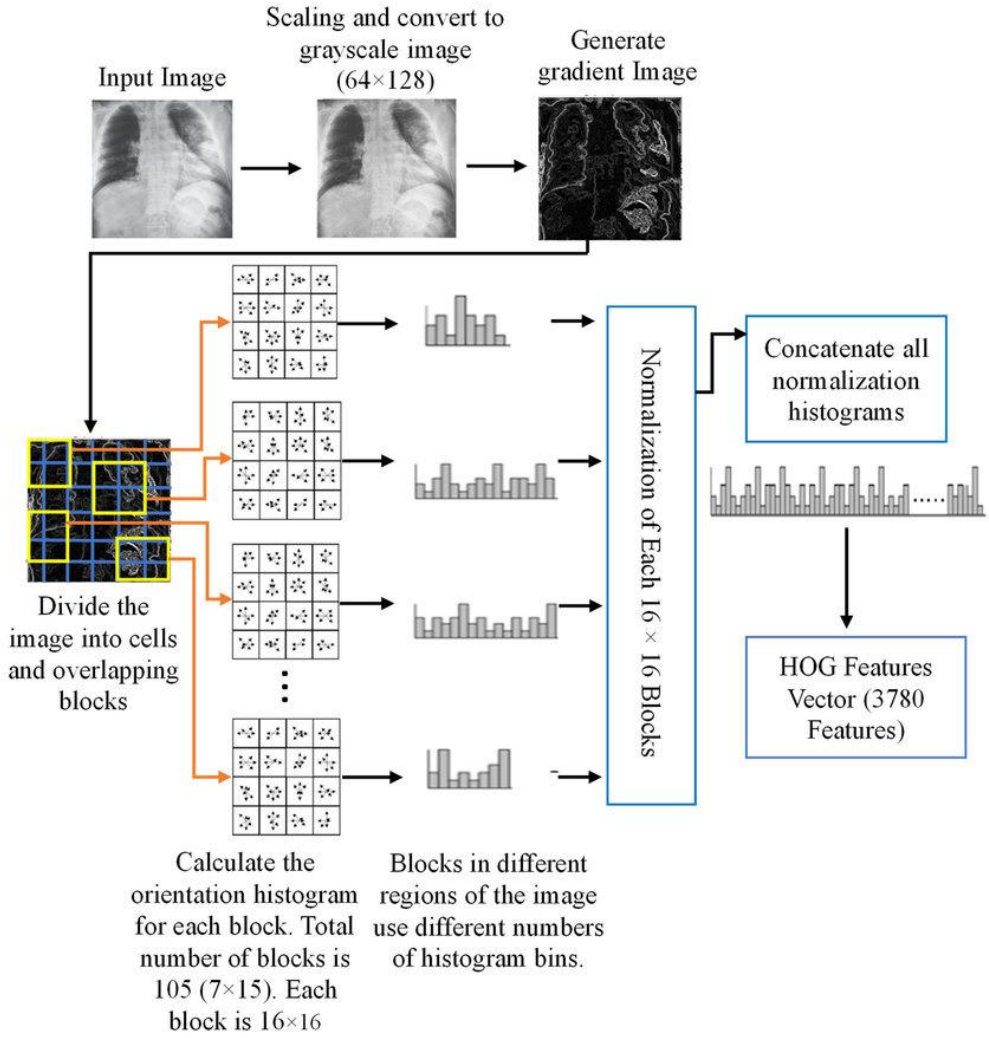


Рисунок 1.5 – Алгоритм роботи HOG

Таким чином, переваги HOG полягають в тому, що він інваріантний до геометричних і фотометричних перетворень, за винятком напрямку руху об'єкта. Груба просторова вибірка, сильна локальна фотометрична нормалізація та точна вибірка спрямування дозволяють ігнорувати рухи тіла, за умови підтримки вертикального положення. Це означає, що описувач HOG особливо корисний для виявляння людей на зображенні.

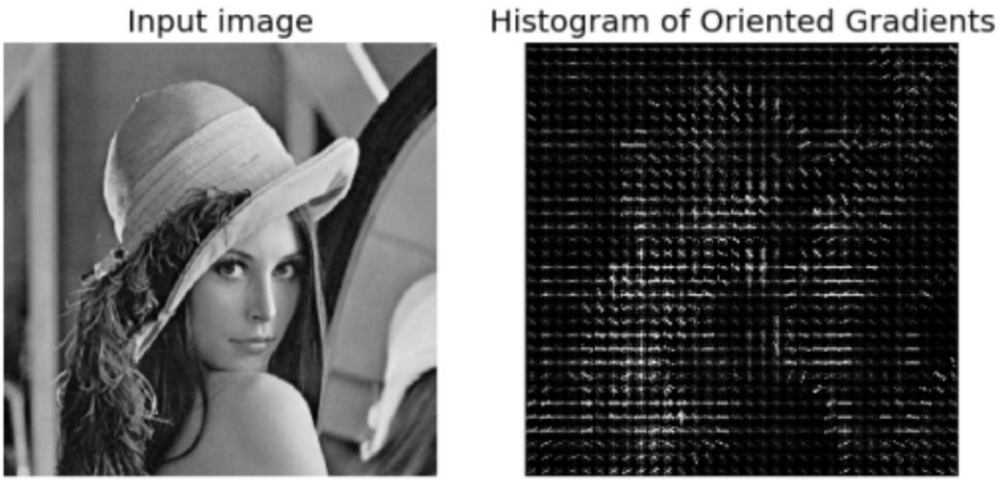


Рисунок 1.6 – Результат обробки зображення HOG

1.2.2. Сучасні методи екстракції ознак

З розвитком глибинного навчання екстракція ознак набула нового імпульсу. Глибокі нейронні мережі автоматично вчаться виявляти ознаки зображень і це призводить до значного підвищення швидкості.

Згорткові нейронні мережі CNN (Convolutional Neural Networks) стали основою більшості сучасних систем для обробки зображень. Вони навчаються на великій кількості даних і автоматично виділяють релевантні ознаки, що робить їх гнучкими та універсальними.

Основними елементами CNN є:

- Згортковий шар: застосування фільтрів для обчислення просторових ознак.
- Шар підвибірки: зменшення розмірів ознак для зниження кількості обчислень і зменшення чутливості моделі до зсувів і шумів.
- Повнозв'язний шар: кінцевий шар для класифікації або іншого завдання.

CNN дозволяє виявляти об'єкти та їх класи у складних умовах, наприклад, при різних освітленнях, кутах зору.

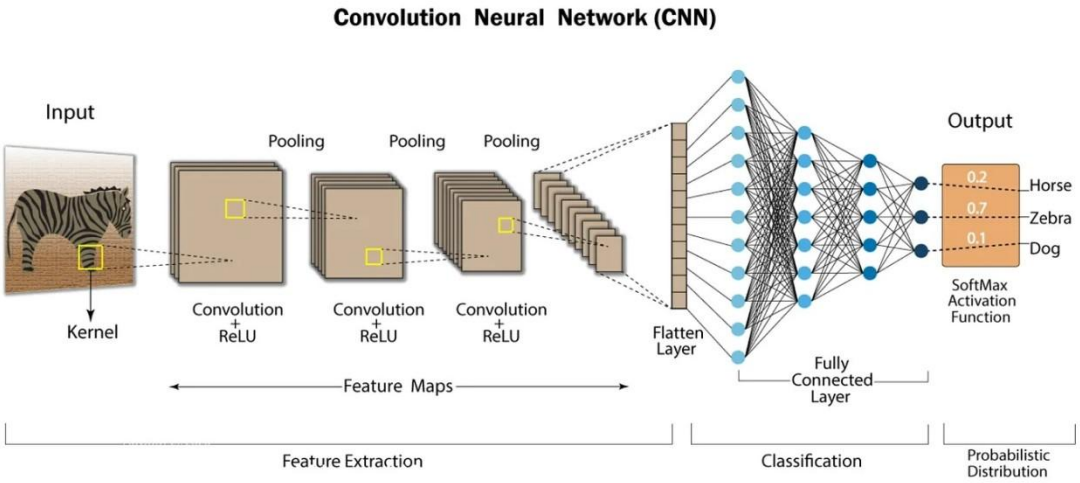


Рисунок 1.7 – Алгоритм роботи CNN

R-CNN(Regions with Convolutional Neural Networks), запропонований Россом Гіршиком у 2014 році, є двоетапною моделлю. Вона поєднала регіональне виявлення з глибокими нейронними мережами. Основними кроками R-CNN є:

- Генерація регіонів-кандидатів, які ймовірно містять об'єкти за допомогою методу селективного пошуку.
- Отримані регіони обробляються згортковою нейронною мережею (CNN) для отримання ознак кожного регіону.
- Для кожного регіону прогнозується ймовірність того, що він містить об'єкт і визначаються координати його меж.
- Для кожного об'єкта проводиться процедура регресії, щоб уточнити положення меж для точнішого визначення об'єкта [11].

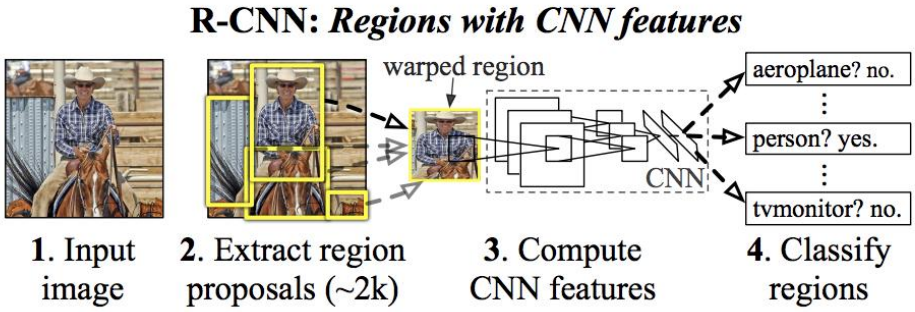


Рисунок 1.8 – Алгоритм роботи R-CNN

Перевагами R-CNN є висока точність і гнучкість, проте значним недоліком вважається низька швидкість роботи. Обробка кожного регіону зображення окремо через CNN є дуже повільною і непридатною для реального часу. Окрім того, для обробки кожного з 2000 регіонів-кандидатів використовуються великі обсяги пам'яті та ресурсів.

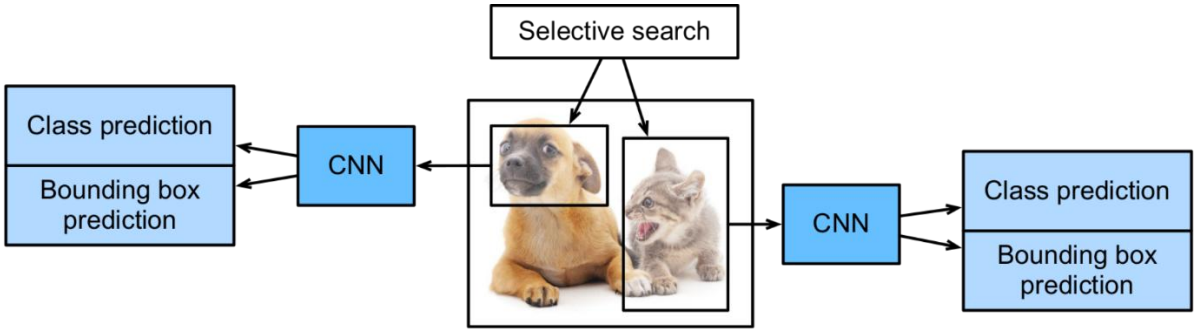


Рисунок 1.9 – Приклад роботи R-CNN

YOLO (You Only Look Once) — це одна з найефективніших моделей для детекції об'єктів у реальному часі. Вона розроблена Джозефом Редмоном і Алі Фархаді у 2016 році та є одним із найпопулярніших методів одночасного виявлення та класифікації об'єктів на зображеннях у реальному часі. YOLO виконує обидві задачі за один прогін нейронної мережі, що робить його особливо швидким. Алгоритм YOLO працює за такими основними кроками:

- Зображення поділяється на сітку, і кожен квадрат цієї сітки прогнозує кілька боксів (bounding boxes), у яких може знаходитися об'єкт.
- Для кожного боксу модель прогнозує координати, ймовірність того, що в боксі знаходиться об'єкт, і до якого класу він належить.

Замість того, щоб спочатку визначати регіони інтересу, а потім їх класифікувати, YOLO робить все одночасно, що значно пришвидшує процес [12].

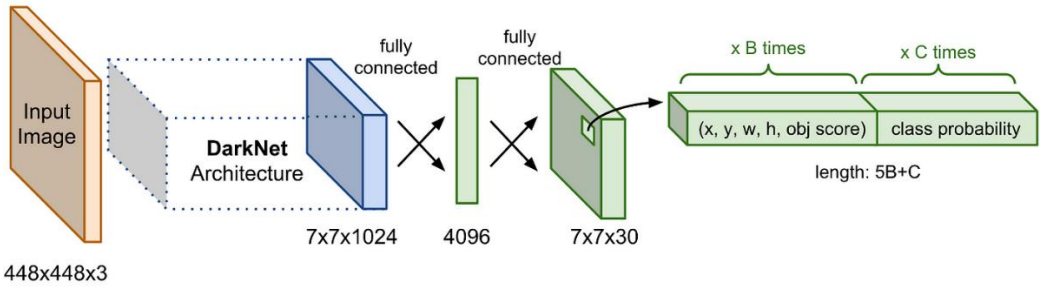


Рисунок 1.10 – Структура YOLO

Перевагами YOLO є висока швидкість, що робить її ідеальною для реального часу та висока точність для простих і чітко окреслених об'єктів. Недоліком є можливі проблеми з дрібними або частково перекритими об’єктами.

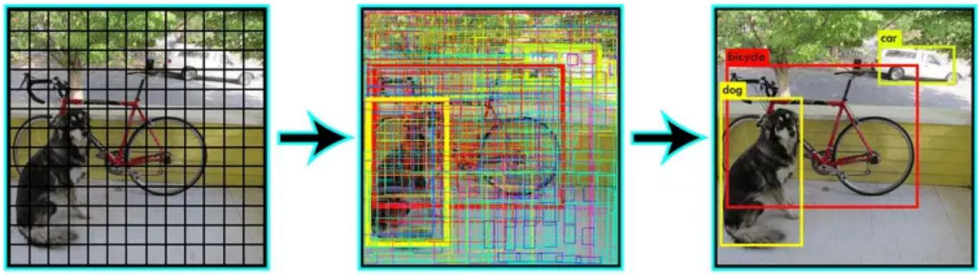


Рисунок 1.11 – Приклад роботи YOLO

1.2.3. Гібридні методи екстракції ознак

Гібридні методи екстракції ознак, які поєднують глибоке навчання з класичними алгоритмами, використовують сильні сторони нейронних мереж для автоматичної екстракції релевантних ознак, але при цьому зберігають ефективність класичних методів для їх подальшого зіставлення або трекінгу. Це дозволяє досягти високої точності і стійкості до різноманітних трансформацій зображень, таких як зміна масштабу, обертання та освітлення.

SuperPoint, розроблений Деніелом ДеТоне, Томасом Маллінасом та Ендрю Рабіновічем, був представлений у 2018 році. Це глибока нейронна мережа для виявлення та опису ключових точок у зображенні. SuperGlue, до роботи над яким був долучений ще Пол-Едуард Сарлос, був представлений у

2020 році. Це алгоритм на основі графових нейронних мереж, який відповідає за пошук відповідностей між точками на різних зображеннях. Вони особливо ефективні у складних умовах, коли необхідно знайти відповідність між зображеннями, зробленими під різними кутами чи в умовах різного освітлення. SuperPoint складається з глибокої конволюційної нейронної мережі (CNN), яка одночасно виконує дві задачі:

- Мережа передбачає наявність ключових точок на зображенні, аналогічно класичним алгоритмам, як SIFT, але використовує глибоку архітектуру для кращої точності.
- Дескриптори генеруються для кожної виявленої ключової точки, що дозволяє використовувати їх для подальшого порівняння між різними зображеннями.

SuperPoint використовує методи навчання без нагляду для побудови ключових точок та їх дескрипторів. Це робить його більш стійким до варіацій в освітленні, масштабі та перспективі, що є важливим для складних реальних застосувань. SuperPoint генерує ключові точки в просторовій сітці, що робить його ефективним навіть для масштабних і динамічних сцен [6].

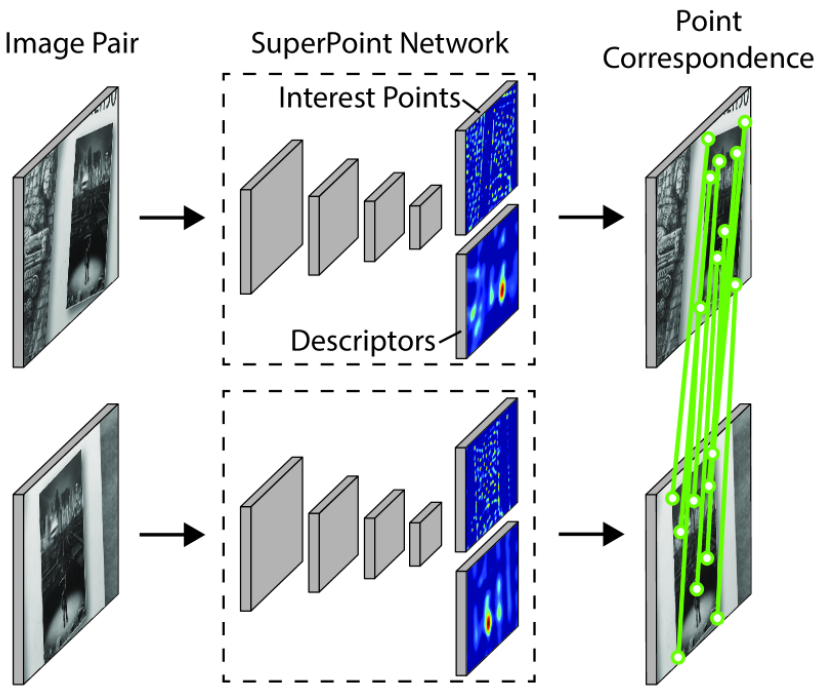


Рисунок 1.12 – Принцип роботи SuperPoint

SuperGlue бере ключові точки та дескриптори, згенеровані SuperPoint або іншим екстрактором ознак. Для співставлення точок використовується GNN, яка будує граф для кожного зображення, де вершини графа — це ключові точки, а ребра представляють зв'язки між ними. SuperGlue порівнює графи двох зображень для знаходження відповідних точок. Після аналізу графів, SuperGlue повертає пари відповідних точок, що використовуються для завдань, таких як побудова гомографії або виявлення об'єктів [13].

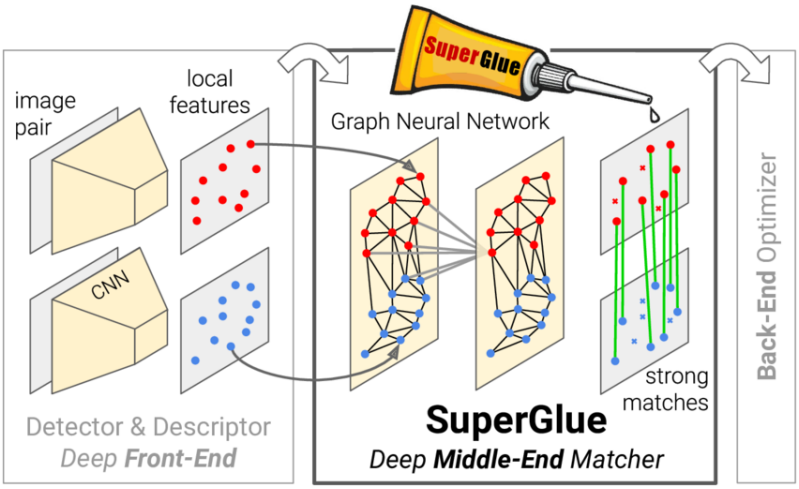


Рисунок 1.13 – Принцип роботи SuperGlue

Перевагами використання SuperPoint і SuperGlue є висока точність в умовах, де традиційні методи, можуть мати труднощі та можливість працювати з великими зображеннями і в умовах змін. Найголовнішими недоліками є певні вимоги до апаратного забезпечення і недостатня швидкість для моментальної роботи.

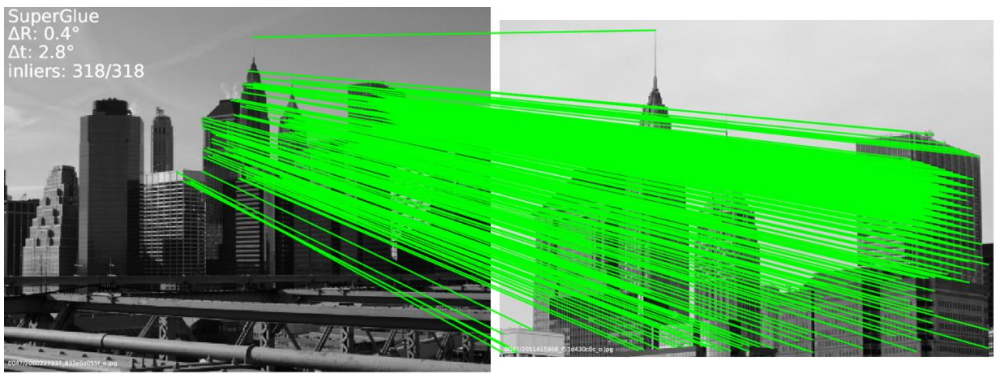


Рисунок 1.14 – Приклад роботи SuperPoint – SuperGlue

D2-Net — це ще один сучасний гібридний метод, запропонований Альяном Дуссером, Володимиром Токмаковим та Паскалем Фуїлем у 2019 році, який використовує глибоку нейронну мережу для одночасного виявлення ключових точок і створення їх дескрипторів. Він базується на ідеї спільного навчання, що дозволяє знаходити ключові точки та їх дескриптори з одного і того ж мережевого виходу. D2-Net ефективно вирішує завдання пошуку і зіставлення ключових точок на зображеннях з великими змінами освітлення, перспективи та масштабу. Основні етапи роботи методу схожі на попередню мережу, але відрізняється тим, що для зіставлення точок між двома зображеннями використовує NN-based matching, який порівнює дескриптори знайдених точок [14].

Перевагами D2-Net є стійкість до змін умов та простіші алгоритми зіставлення точок. Недоліками є високі обчислювальні витрати та повільніший матчинг. Використання NN-based matching може бути менш оптимізованим порівняно з методами, що застосовують спеціалізовані рішення для прискорення пошуку відповідностей, як, наприклад, SuperGlue.

Рисунок 1.15 – Приклад роботи D2-Net – NN-based matching

1.3. Критерії ефективності пошуку об’єкта інтересу на зображенні

Ефективність пошуку об'єкта інтересу на зображенні визначається рядом показників, які дозволяють оцінити, наскільки точно, швидко та надійно алгоритм може знайти об'єкти в різних умовах. Ці критерії враховують як

якість виявлення і зіставлення ключових точок, так і загальну продуктивність системи в реальних умовах, таких як змінне освітлення, перспективні спотворення, масштаб та наявність шумів [8].

Основні критерії ефективності екстракції ознак і їх відповідності для пошуку об'єктів на зображеннях включають такі метрики:

- Точність виявлення (Accuracy): відсоток правильних відповідностей між ключовими точками об'єкта на різних зображеннях.
- Час обробки (Processing Time): швидкість виконання алгоритму. Критерій особливо важливий для пошуку об'єктів в реальному часі на зображеннях з дронів або камер.
- Стійкість до трансформацій (Robustness): здатність алгоритму правильно виявляти та зіставляти ознаки в умовах змін масштабу, обертання, освітлення і часткової зміни об'єкта.
- Кількість ключових точок (Number of Keypoints): кількість виділених ключових точок на зображенні.
- Стійкість до фону (Background Insensitivity): чутливість методу до території, на якому знаходиться об’єкт.
- Обчислювальна складність (Computational Complexity): загальний ресурс, необхідний для роботи алгоритму. Це включає використання пам'яті, CPU та GPU.

Таблиця 1.1 – Порівняння ефективності методів пошуку об’єкта інтересу

Метод	Точність виявлення	Час обробки	Стійкість до трансформацій	Кількість ключових точок	Стійкість до фону	Обчислювальна складність
SIFT	Висока	Високий	Висока	Середня	Середня	Висока
SURF	Середня	Середній	Середня	Середня	Середня	Середня
HOG	Низька	Низький	Низька	Низька	Низька	Низька
YOLO	Висока	Низький	Середня	Немає	Середня	Середня
R-CNN	Дуже висока	Дуже високий	Середня	Немає	Середня	Середня
SuperPoint SuperGlue	Дуже висока	Високий	Дуже висока	Висока	Середня	Висока
D2-Net	Висока	Високий	Висока	Висока	Висока	Висока

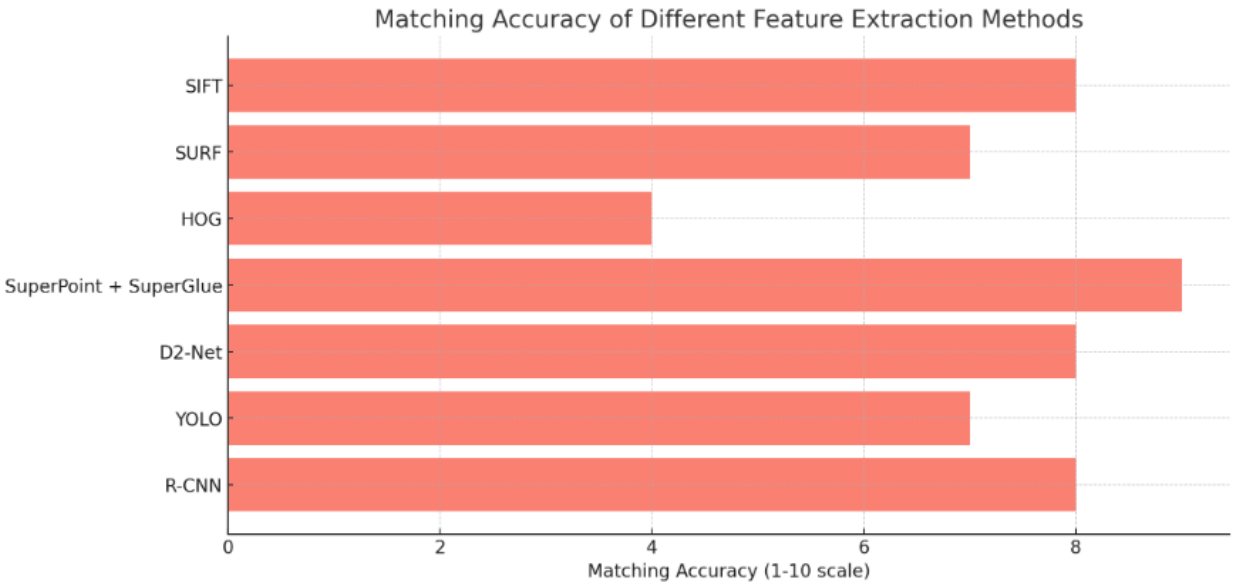


Рисунок 1.16 – Графік порівняння точності екстрації ознак методів

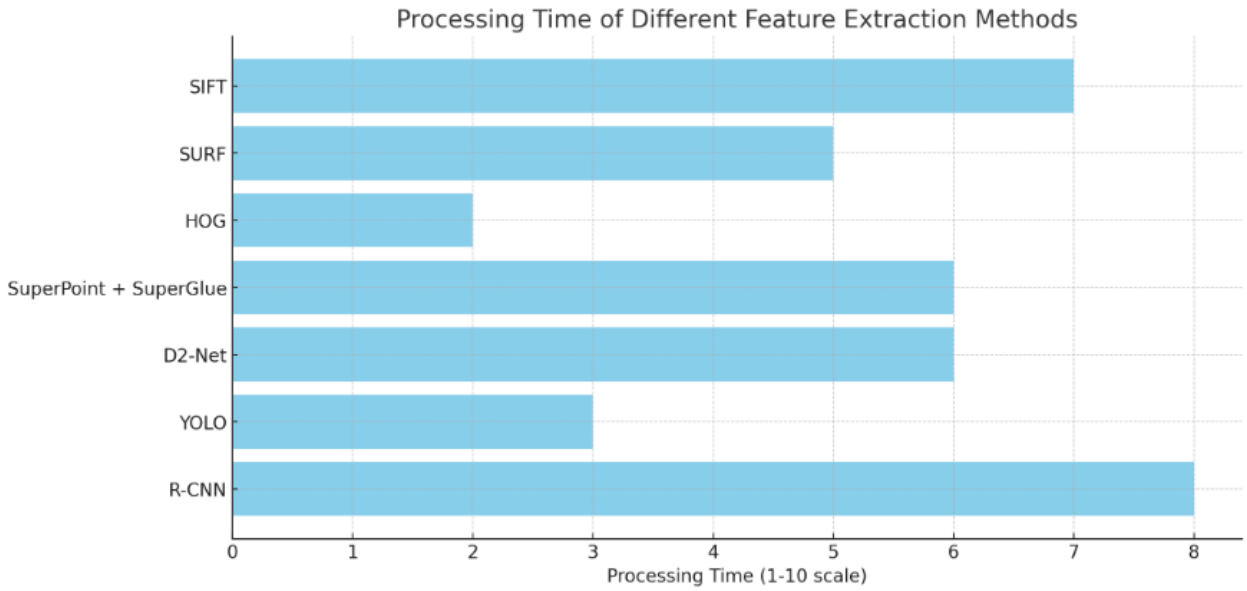


Рисунок 1.17 – Графік порівняння часу обробки методів

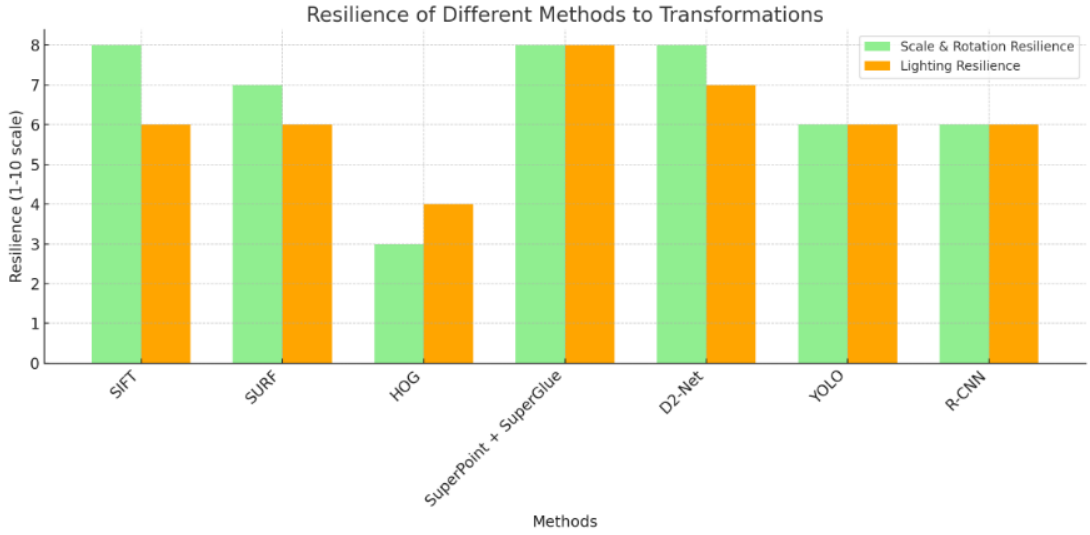


Рисунок 1.18 – Графік порівняння стійкості до трансформацій методів

Для аналізу продуктивності розглянутих методів виявлення та зіставлення об'єктів були порівняні: кількість обчислень (FLOPs), точність (Accuracy) та час обробки (Inference Time).

Таблиця 1.2 – Порівняння продуктивності методів пошуку об’єкта інтересу

Метод	FLOPs (G)	Accuracy	Inference Time (мс)	Висновок
SIFT	~3.5	Висока	120-150	Обчислювально важкий, повільний
SURF	~2.8	Середня	60-80	Швидший за SIFT, але менш точний.
HOG	~1.5	Низька	20-50	Ефективний для простих об'єктів, обличь.
YOLOv5	~16	Висока	5-15	Баланс швидкості та точності.
R-CNN	~250	Висока	100-200	Точний, але повільний.
SuperPoint	~1.3	Висока	3-5	Точний та швидкий
D2-Net	~20	Висока	50-100	Орієнтований на щільний опис ознак.

1.4. Постановка завдання

Сучасна технологія безпілотних літальних апаратів стрімко розвивається в цивільному та військовому контекстах протягом останніх чотирьох десятиліть. БПЛА являють собою перетин двох важливих тенденцій у військовій сфері: точного характеру зброї та дистанційного керування без ризику для пілота. Безпілотники, що використовуються в сучасній війні,

змінили пропонують унікальні тактичні переваги та підвищують оперативну ефективність у різних бойових сценаріях [15]. Військові БПЛА використовуються для різних цілей:

- Спостереження;
- Розвідка;
- Цілеспрямовані удари.

Безпілотники класифікуються на основі різних характеристик, як: функція, розмір, навантаження, географічний діапазон, тривалість та висота польоту [16]. Відповідно до класифікації НАТО, перелік безпілотників в Україні демонструє I та III класи. Під час сучасної війни малі військові безпілотники, інтегровані з наземними підрозділами, зазвичай використовуються для цілевказання, спостереження, оцінки бойових ушкоджень та інформаційної війни [17].

Проте, незважаючи на значний прогрес у технологіях безпілотників, існує ряд викликів, що потребують подальшого вдосконалення. Програми для дронів повинні працювати максимально швидко, точно й не залежити від високих обчислювальних здатностей машин, щоб забезпечити ефективне виконання завдань у реальному часі в динамічних бойових умовах.

Отже, метою роботи є розробка програми, що дозволить військовому дрону автоматично розпізнавати ворожу техніку на зображеннях місцевості в реальному часі, використовуючи попередньо завантажений зразок об’єкта. Система повинна бути стійкою до змін ракурсу, освітлення, масштабу та часткового перекриття об'єкта. Критично важливими будуть низький час обробки зображення та висока точність розпізнавання.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- Попереднє завантаження зображення об'єкта ворожої техніки, який буде шукати дрон;
- Розпізнавання, локалізація та визначення класу об'єкта (об’єктів) на поточному зображенні;

- Оптимізація поточного зображення для збільшення швидкості подальшої обробки шляхом вирізання об’єктів з фону;
- Порівняння об'єкту на шаблонному зображенні з об’єктом (об’єктами) на поточному для виявлення потрібного;

Формалізація задачі:

1. Вхідні дані:
    - Зображення-зразок з об'єктом, який необхідно виявити;
    - Поточний кадр з дрона.
2. Вихідні дані:

- Координати знайденого об’єкта (об’єктів);
- Впевненість;
- Клас знайденого об'єкта (об’єктів);
- Кількість співпадінь виявленого об'єкта (об’єктів) і шаблону;
- Зображення шаблону та поточного кадру з найбільшою кількістю співпадінь;

Таким чином, програма забезпечуватиме швидке та надійне розпізнавання об’єкту на зображенні з дрону.

2. ВИБІР МЕТОДІВ РЕАЛІЗАЦІЇ

2.1. Обрані моделі пошуку і відстеження об'єктів інтересу на зображенні місцевості

Для вирішення завдання виявлення і відстеження об'єктів інтересу на зображенні місцевості військовими дронами, були обрані наступні методи: YOLO, SuperPoint і SuperGlue. Ці інструменти ефективно поєднують можливості точного й швидкого виявлення об’єктів у реальному часі, що є важливими критеріями у військовій сфері.

2.1.1. YOLO

YOLO (You Only Look Once) – алгоритм, здатний виявляти об’єкти в один етап, виконуючи виявлення та класифікацію одночасно. Він був обраний через його високу швидкість обробки, точність, стійкість до змін та здатність працювати з великою кількістю класів об’єктів. YOLO вважається ефективнішим за багато інших алгоритмів для визначення об'єктів.

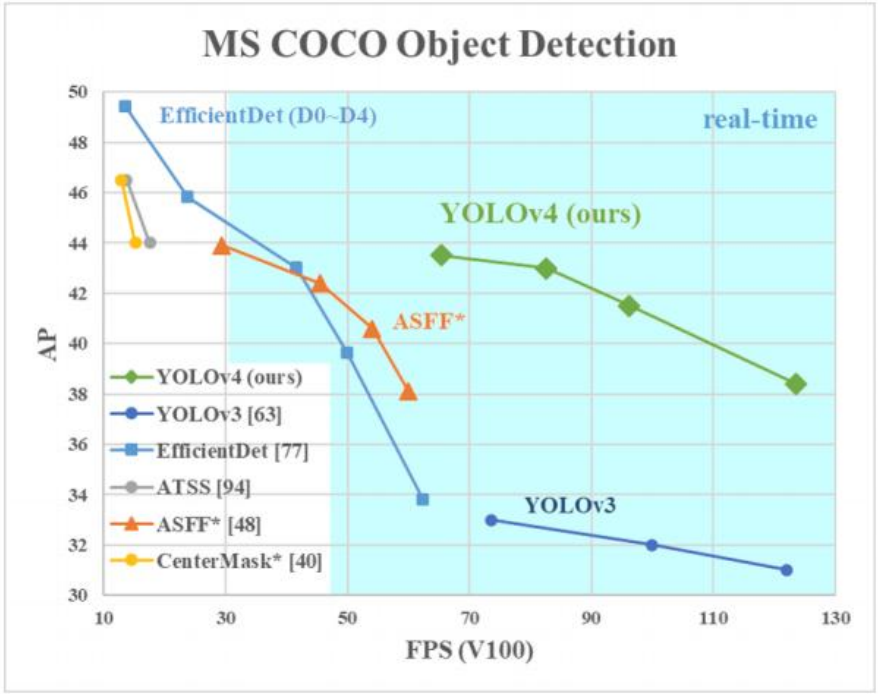


Рисунок 2.1 – Ефективність методів детекції об’єктів

Перевагою YOLO є те, що цей метод не має проблем двохрівневих архітектур, а саме: низька швидкість та розгляд зображення по частинам. Архітектура перших блоків YOLO, насправді, мало чим відрізняється від інших описувачів ознак. Спочатку на вхід надходить фотографія, далі за допомогою CNN (в YOLO це Darknet-53) створюються feature maps, потім вони аналізуються, і на виході надаються розміри і позиції bounding boxes та класи, яким вони належать. Останнім блоком, який саме дозволяє YOLO вирватися в лідери ефективності, є Dense Prediction.

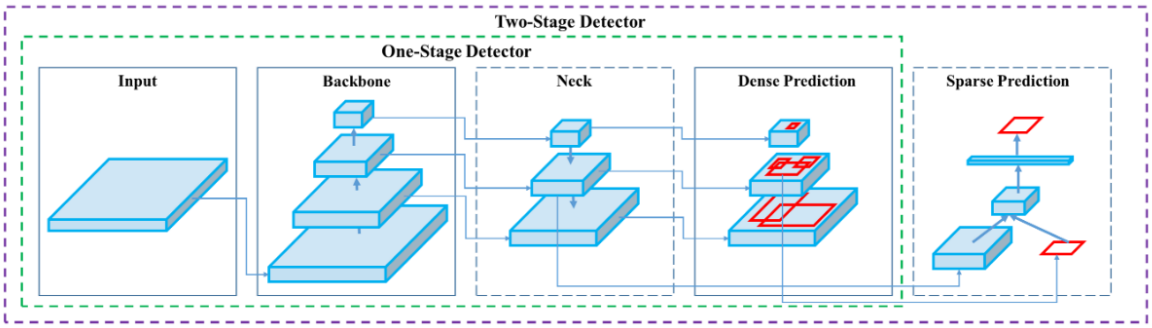


Рисунок 2.2 – Різниця архітектур YOLO та двохрівневих методів

Блоки архітектури детекторів після створення feature maps працюють таким чином:

- Neck – це блок, який об'єднує інформацію з окремих шарів попереднього блоку для підвищення точності прогнозування.
- Sparse Prediction – це блок дворівневих алгоритмів, який окремо визначає області і потім їх класифікує.
- Dense Prediction – блок, який дозволяє однорівневим алгоритмам одночасно передбачати класи і координати об'єктів по всій площині зображення, уникаючи етапу генерації регіонів [18].

Алгоритм навчається таким чином:

- Зображення розбивається на фіксовану сітку (в YOLO зазвичай 13x13);

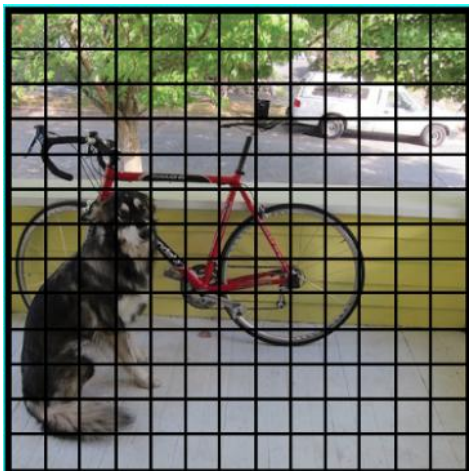


Рисунок 2.3 – Етап поділу зображення на сітку

- Навколо клітин-якорей рисуються прямокутники визначення об'єкта «bounding-box» різних форм. Їх висота, ширина і положення, розраховуються відносно центру комірки за один раз. Вони малюються за технікою «anchor boxes», які задаються на початку користувачем, або вже є в датасеті для тренування.

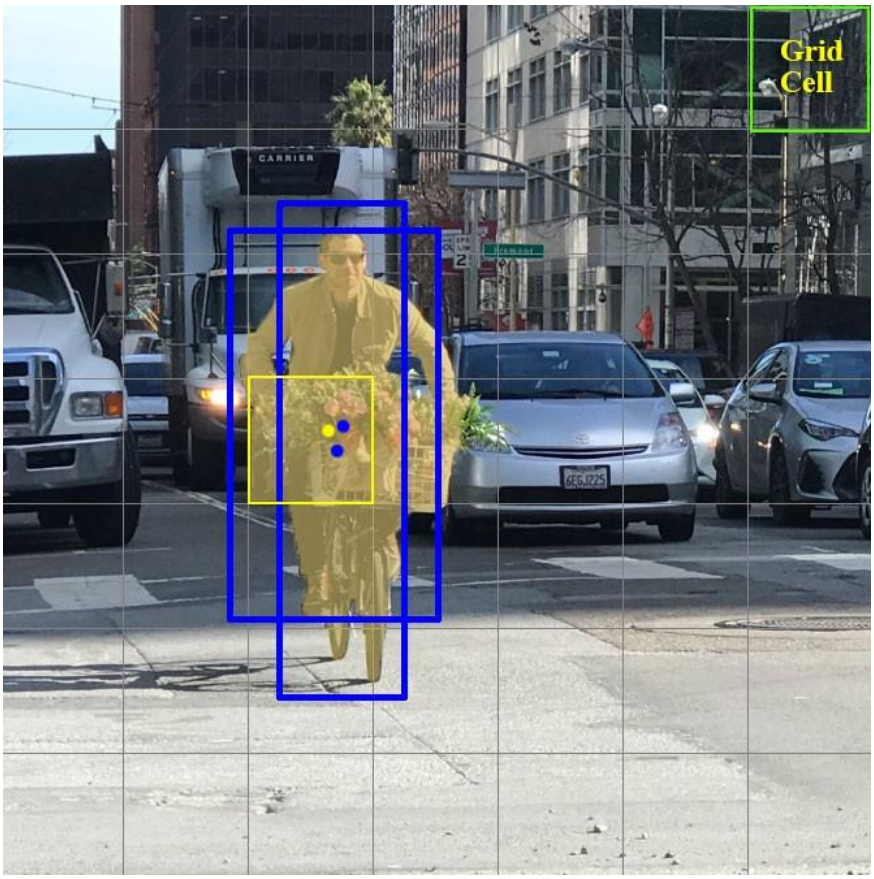


Рисунок 2.4 – Етап відрисовки bounding-box

- Зображення з набору даних обробляється нейронною мережею (на додаток до картинки в тренувальному датасеті повинні бути визначені фактичні позиції і розміри bounding boxes для об'єктів на ній).
- У вихідних даних визначаються дві основні речі:

  1. Який з anchor boxes є найбільш підходящим і як його можна відредагувати так, щоб він вписував об'єкт найкраще;

  2. Чи існує об’єкт всередині цього anchor box і що він собою являє.

Для кожної клітини отримуються розташування anchor boxes, їх якість і ймовірність належності до кожного класу.

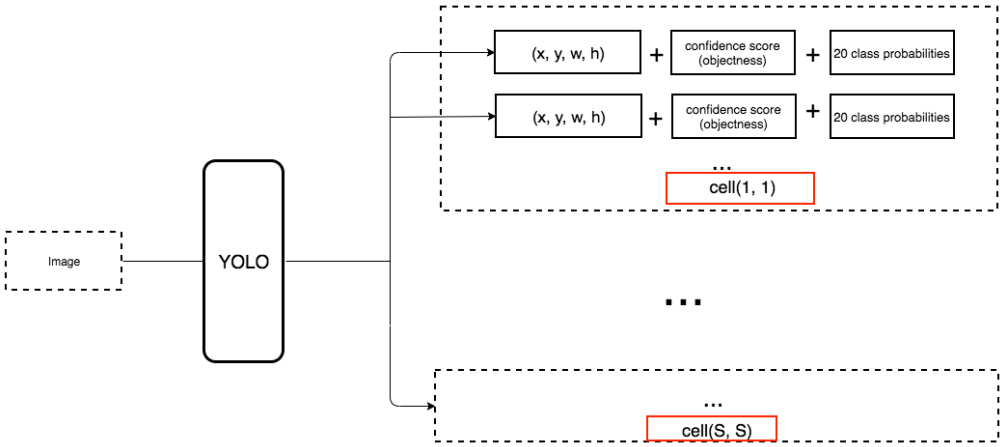


Рисунок 2.5 – Дані, отримані на виході

Якість клітин якоря визначається за допомогою метрики IoU під час навчання таким чином:

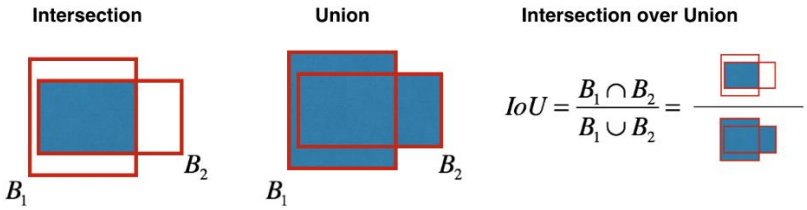


Рисунок 2.6 – Робота метрики IoU для визначення якості якоря

- Це значення використовується для розрахунку загального показника достовірності confidence score (впевненості в тому, що саме потрібний об'єкт розташований всередині передбачуваного прямокутника). Це фільтр, що дозволяє повністю виключити неточні прогнози.
- Коли залишаться тільки bounding boxes з високим confidence score, передбачення можуть виглядати приблизно ось так:

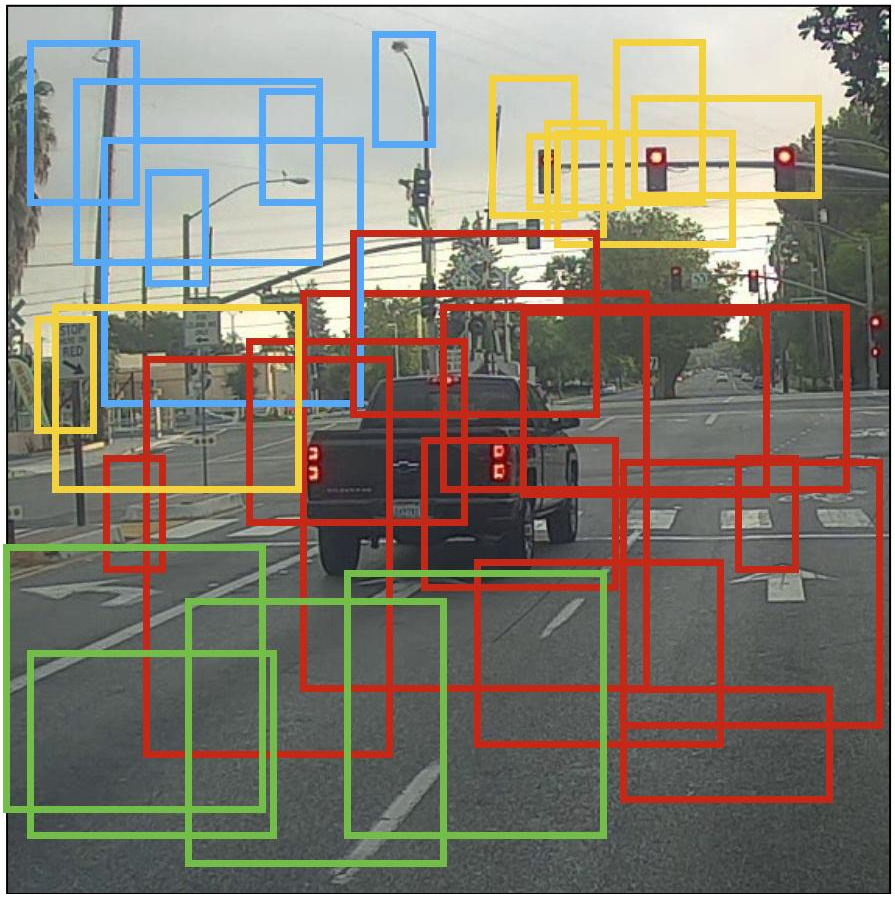


Рисунок 2.7 – Передбачені bounding boxes з високим confidence score

- Для фільтрації bounding boxes таким чином, щоб для одного об’єкта існував тільки один передбачений bounding box, використовується метод NMS (non-max suppression).

Рисунок 2.8 – Фільтрація bounding boxes

- В результьаті отримуються: точні координати bounding box, клас об'єкта та коефіцієнт впевненості, що вказує на ймовірність правильності детекції [18].

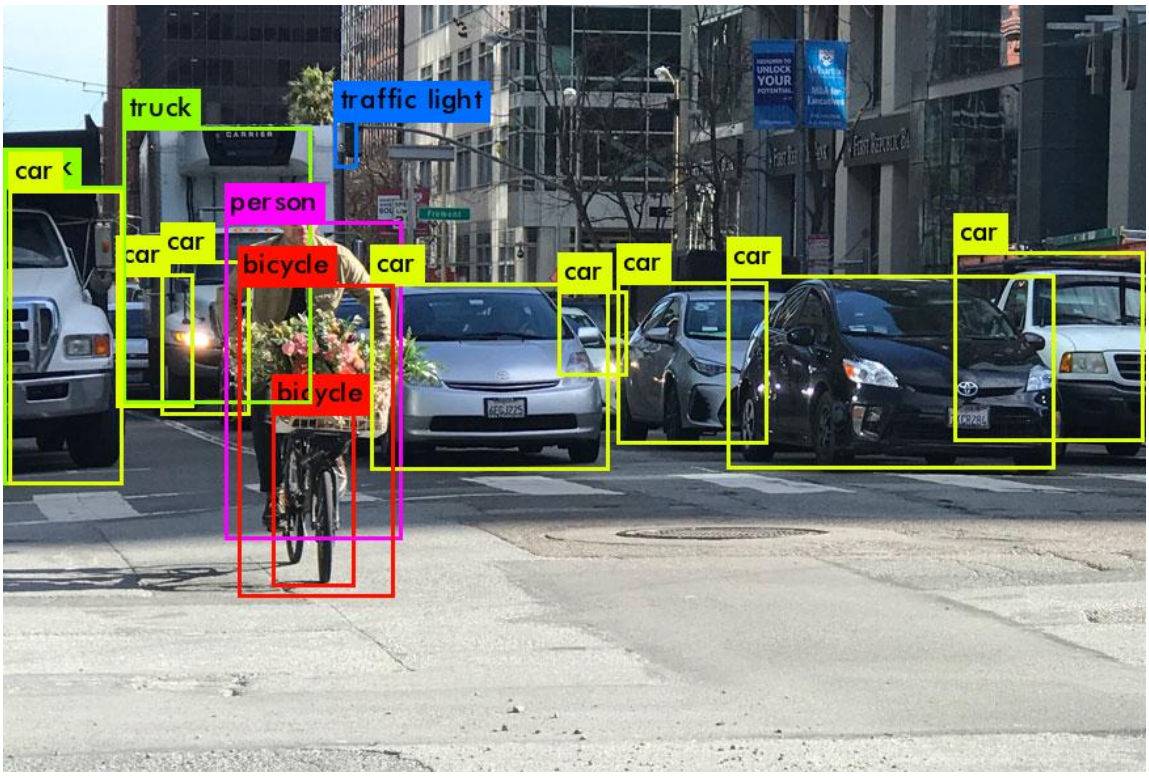


Рисунок 2.9 – Результат роботи YOLO

Отже, архітектура YOLO побудована таким чином, щоб забезпечити баланс між швидкістю і точністю, завдяки паралельному виконанню детекції

й класифікації об'єктів. Для реалізації поставленої задачі, обрана версія YOLO v5s, навчена на базі COCO (Common Objects in Context), завдяки її ефективності у виявленні об'єктів ворожої техніки з військових дронів і високій підтримці на різних видах пристроїв.

2.1.2. SuperPoint та SuperGlue

SuperPoint та SuperGlue – це сильна комбінація методів для виявлення та зіставлення ключових точок між зображеннями. SuperPoint виконує функцію екстракції ключових точок, а SuperGlue – їх високоточного зіставлення. Вони були обрані для даної роботи завдяки високій точності та надійності в умовах зміни перспективи, масштабу і варіацій освітлення, що є ключовими вимогами для роботи в реальних умовах на військових дронах.

SuperPoint – це нейронна мережа, натренована на виявлення і дескрипцію ключових точок на зображеннях без зовнішнього нагляду, використовуючи self-supervised навчання, що і виділяє його серед конкурентів. Метод навчається шляхом створення штучних варіацій зображень (афінні трансформації, зміна яскравості і контрасту) та генерує пару оригінал-варіація. В результаті порівняння loss обчислюється між вихідними і перетвореними точками та дескрипторами, дозволяючи моделі розвивати інваріантність до зміщень та змін освітлення.

Основні елементи архітектури SuperPoint включають:

- Pre-trained Convolutional Backbone, що використовується як основа для обробки вхідного зображення.
- Encoder, який обчислює heatmap з вірогідностями наявності ключових точок у певних частинах зображення.
- Descriptor Head, який представляє унікальні характеристики точки для подальшого зіставлення з іншими зображеннями.

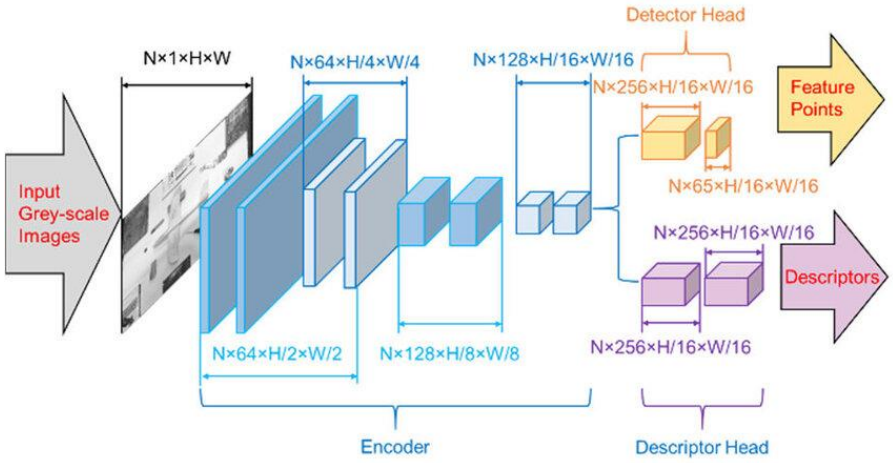


Рисунок 2.10 – Архітектура SuperPoint

SuperPoint працює за такими кроками:

- За допомогою Pre-trained Convolutional Backbone обчислюються початкові ознаки зображення і створюються feature maps. Зазвичай це декілька згорткових шарів (наприклад, ResNet або подібні CNN), які обчислюють просторові особливості. Backbone тренується на великому наборі зображень, щоб навчитися витягати корисні ознаки і вже має добре розвинуті представлення для різних патернів у зображеннях.
- За допомогою Encoder створюється карта heatmap, яке проектує отримані feature maps у форму, що вказує ймовірність наявності ключових точок у кожному пікселі.
- Після створення карти, на основі локальної максимізації у кожній клітині виділяються лише найсильніші точки. Це зменшує кількість ключових точок і підвищує точність.
- За допомогою Descriptor Head feature maps проектуються у векторний простір, де кожна точка має вектор для визначення її унікальності.
- Кожен дескриптор нормалізується, щоб бути інваріантним до масштабу і орієнтації, дозволяючи точкам бути порівняними між різними зображеннями [19].

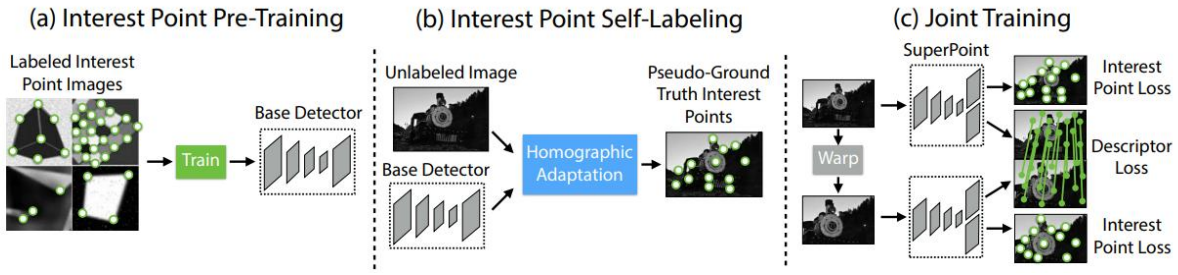


Рисунок 2.11 – Етапи роботи SuperPoint

SuperGlue – це нейронна мережа, розроблена для співставлення ключових точок між зображеннями на основі аналізу просторових і контекстуальних зв'язків між точками, виявленими попередніми методами, такими як SuperPoint. Використовуючи attention механізми і самостійне навчання, SuperGlue здатний навчатися без залежності від традиційних геометричних методів, таких як RANSAC, що підвищує точність у складних умовах [13].

Основні елементи архітектури SuperGlue включають:

- Encoder, що обробляє ключові точки і їхні дескриптори, формуючи початкові представлення для кожної точки. Він приймає інформацію, отриману за допомогою SuperPoint, об'єднує дані і створює початковий векторний опис.
- Self-Attention, який дозволяє кожній ключовій точці враховувати інформацію про інші точки в межах одного зображення. Це покращує локальний контекст кожної точки, враховуючи її положення відносно інших.
- Cross-Attention, який здійснює взаємодію між точками з двох різних зображень. Він дозволяє кожній точці в одному зображенні отримувати інформацію про точки іншого зображення, що підвищує ймовірність знаходження відповідностей між схожими точками на обох зображеннях [20].

- Sinkhorn Layer, який реалізує алгоритм нормалізації Сінкгорна для забезпечення двонаправленої відповідності між точками. Це дозволяє забезпечити, що кожній точці зіставлено лише одну відповідну точку [21].
- Matching Layer, який завершує процес шляхом формування остаточних відповідностей для кожної ключової точки на основі їхньої подібності. Це дозволяє відсіювати неправильні відповідності та залишати лише найвищі.

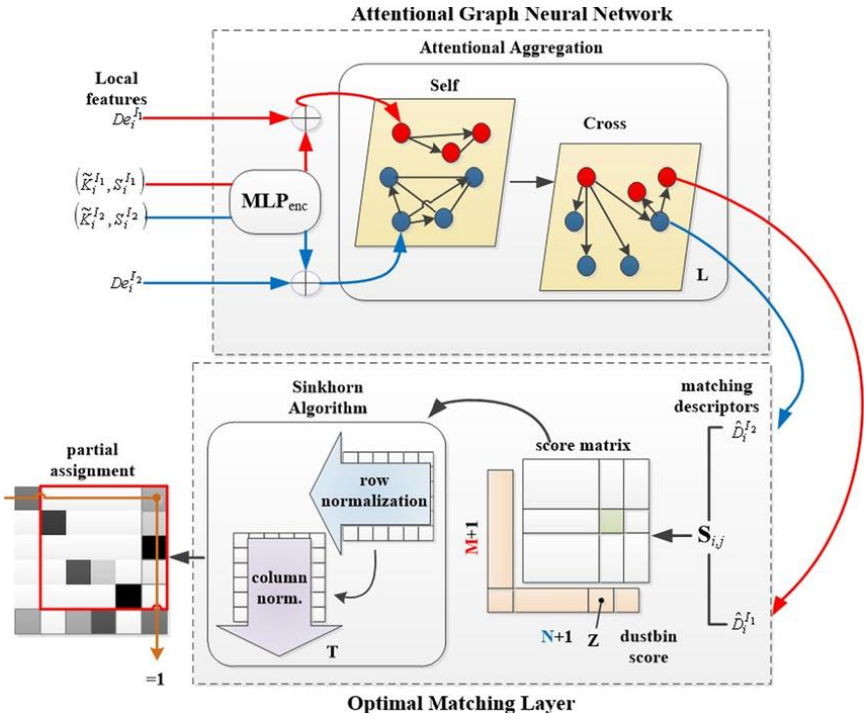


Рисунок 2.12 – Архітектура SuperGlue

SuperGlue працює за наступними кроками:

- З кожного зображення формується граф точок, де вершинами є ключові точки, визначені за допомогою SuperPoint, а ребрами – відстані або інші показники схожості.
- Ключові точки і їхні дескриптори проходять через кодувальну мережу, яка обробляє їхні просторові та дескрипторні дані, створюючи початкове представлення для подальшої обробки.

- Механізм уваги Self-Attention обчислює подібності між точками з урахуванням усіх сусідніх точок, забезпечуючи локальні та глобальні відповідності одночасно.
- Cross-Attention використовується для встановлення взаємозв'язків між точками на різних зображеннях. Це дозволяє кожній точці зіставлятися з найбільш схожими точками на іншому зображенні.
- Алгоритм нормалізації Сінкгорна застосовується для регуляризації матриці відповідностей. Це дозволяє забезпечити двонаправлену відповідність, при якій кожна точка може бути співставлена лише з однією точкою або не співставлена зовсім.
- На основі відфільтрованих відповідностей формується кінцевий список пар точок між двома зображеннями, які мають найвищий рівень відповідності. Відсіюються слабкі відповідності, щоб забезпечити точність знайдених пар [13].

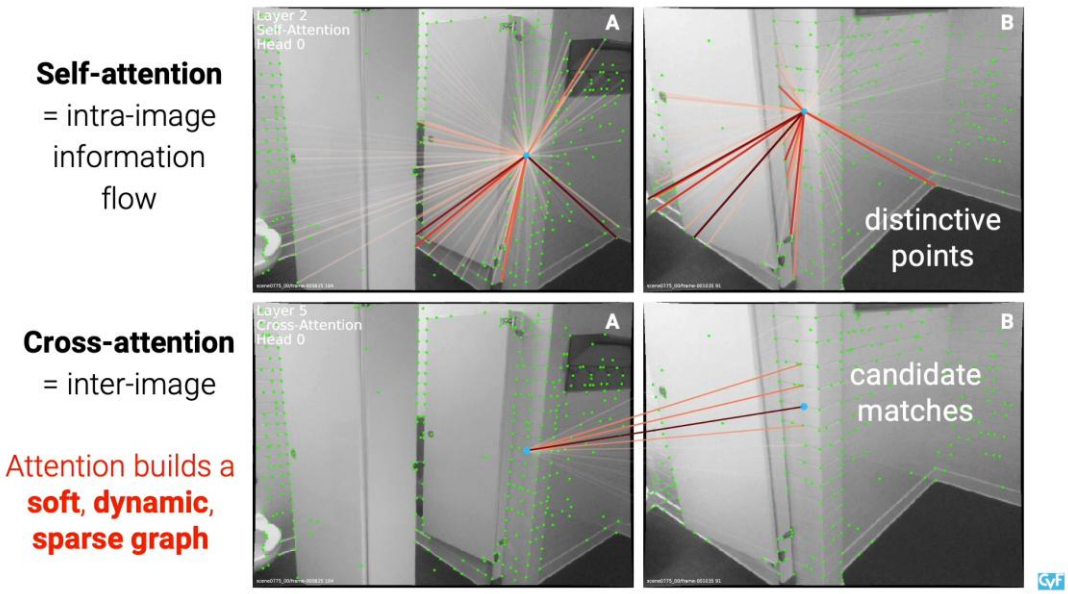


Рисунок 2.13 – Attention етапи роботи SuperGlue

Отже, комбінація SuperPoint і SuperGlue ідеально підходить для виявлення дроном ворожої техніки за зразком, оскільки забезпечує стійкі та точні визначення ключових точок і співставлення між ними на зображеннях.

Це допомагає збільшити ймовірність розпізнавання саме потрібного об’єкта при складних умовах поточного кадру.

2.2. Інтеграція методів у єдину систему

Для вирішення задачі відстеження і детектування об’єктів ворожої техніки на місцевості розвідувальним БПЛА була створена технологія, яка об’єднує описані вище методи комп’ютерного зору у єдину систему. Саме робота сучасних інструментів в комбінації дозволяє досягти високої точності, ефективності й стабільності в умовах реального часу.

Програма працює за такими основними етапами:

- Завантажується зображення-шаблон з ворожою технікою, яку має виявити розвідувальний дрон та поточне зображення кадру з безпілотнику.
- За допомогою детектору YOLO виявляються об'єкти, їх характеристики та обрізається зайвий фон, що значно спрощує задачу подальшого оброблення. Це дозволить уникнути додаткового навантаження на обчислювальну систему, обробляючи тільки важливі ділянки кадру.
- SuperPoint та SuperGlue виконують зіставлення ключових точок об’єктів на шаблонному і поточному зображеннях, щоб забезпечити впевненість виявлення саме потрібної ворожої техніки за умов наявності інших об’єктів, можливих змін перспективи, освітлення та інших перешкод.
- Отримуються результати у вигляді списку об’єктів, їх співпадінь з шаблоном та зображення, яке демонструє виявлений необхідний об’єкт.

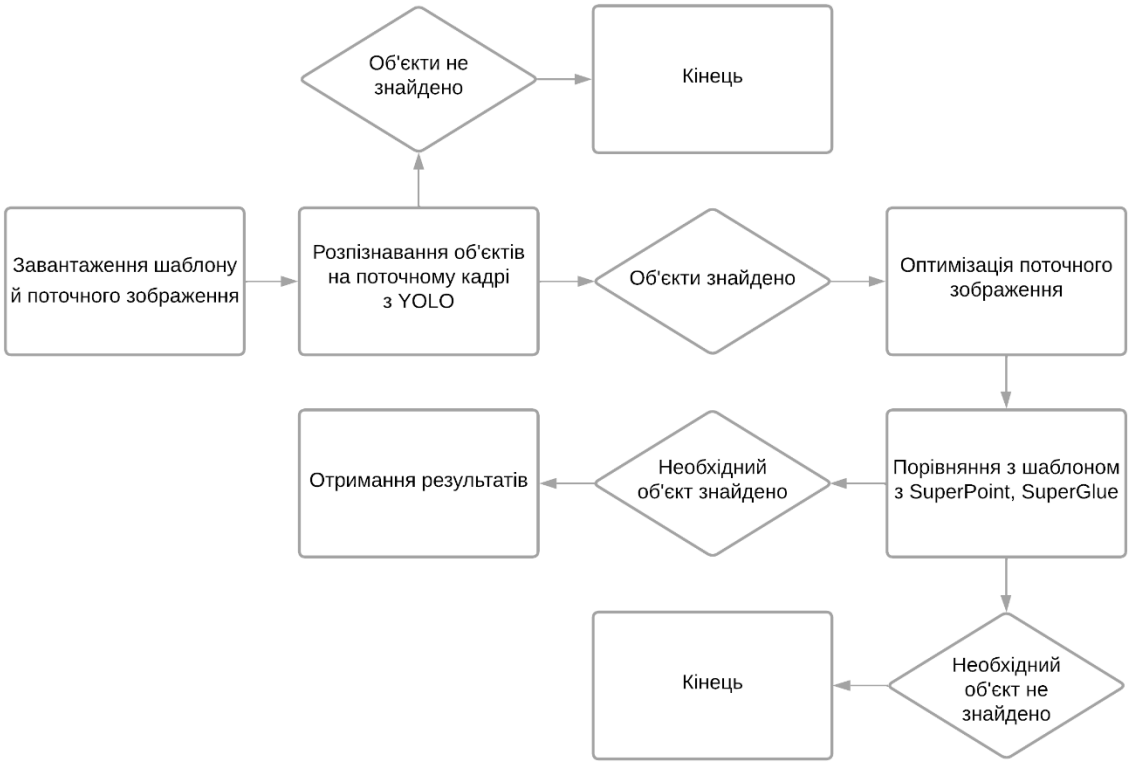


Рисунок 2.14 – Логіка роботи програми

Отже, завдяки інтеграції кількох сучасних методів в єдину систему, нова технологія поєднує в собі найкращі сторони різних інструментів детектування об’єктів. Унікальність технології полягає в тому, що замість того, щоб покладатися на один метод для всього обсягу роботи, вона ефективно об'єднує виявлення об’єктів за допомогою YOLO та точне зіставлення зображень із SuperPoint-SuperGlue. Таке рішення забезпечує не лише ефективне використання ресурсів, а й створює стійку, комплексну систему, що здатна адаптуватися до реальних умов та відстежувати потрібні об’єкти у складних швидко-змінюваних сценах.

3. ІНФОРМАЦІЙНЕ І ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1. Формування вхідних даних

Для тестування роботи програми необхідно підготувати й налаштувати певні вхідні дані. Спершу, підбираються два зображення, які містять однаковий об’єкт, який потрібно відстежити. Для ефективного тестування програми, техніка має бути з різними перспективами, освітленням та фоном. Шаблонне зображення має містити потрібний об’єкт крупним планом. Поточне фото повинно бути реальним кадром з дрону, де можлива велика кількість зайвих об’єктів та фону. Зображення мають мати однакову роздільну здатність й формат.

Рисунок 3.1 – Приклад вхідних зображень шаблону і поточного фото

Далі, необхідно зареєструвати, або увійти в існуючий Google аккаунт і завантажити зображення на Google Drive. Шаблон має називатись «template», а поточне фото – «current».

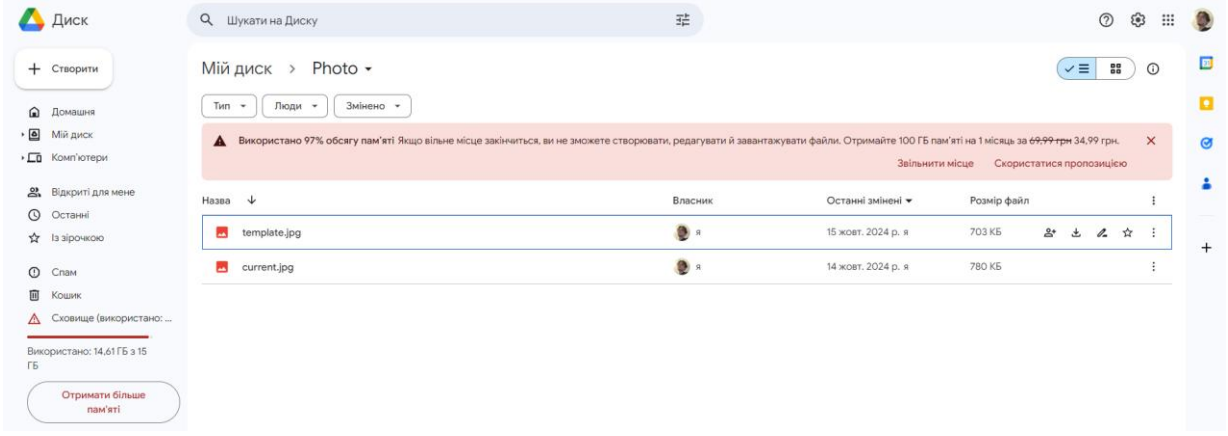


Рисунок 3.2 – Завантаження вхідних зображень на Google Drive

Потім, для коректного відпрацювання SuperPoint, SuperGlue в складних умовах, можна відредагувати їх параметри. Налаштування для SuperPoint задаються у вигляді словника, де input_shape вказує на форму вхідних зображень, а keypoint_threshold – на поріг, за яким визначаються ключові точки. Для SuperGlue задається параметр match_threshold, що визначає допустимий поріг для співпадінь.

Рисунок 3.3 – Налаштування параметрів SuperPoint, SuperGlue

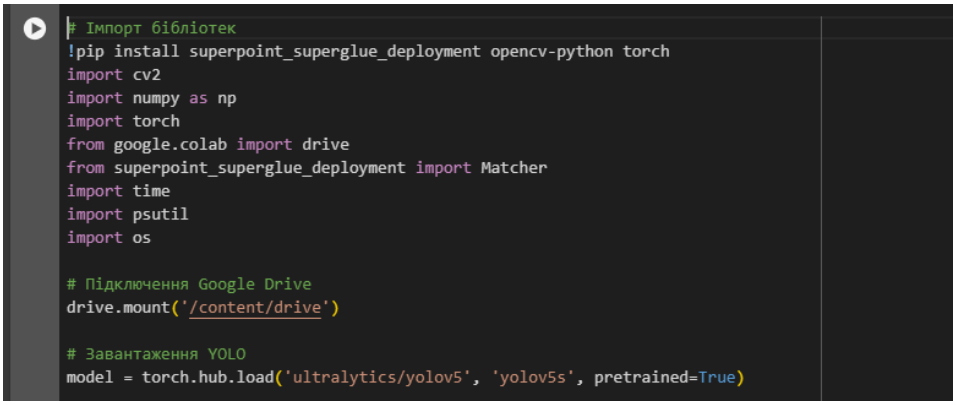
Таким чином, після цих дій, програма буде повністю готовою до тестування.

3.2. Опис програмної реалізації

Для реалізації програми була обрана мова програмування Python. Вибір був обумовлений його простотою, зручністю та широкою підтримкою бібліотек для машинного навчання, комп'ютерного зору та обробки зображень. Python дозволяє швидко розробляти прототипи та реалізовувати складні алгоритми завдяки зрозумілому синтаксису та великій спільноті, що надає підтримку та ресурси.

Середовищем виконання став Google Colab. По-перше, він дозволяє безкоштовно використовувати потужні апаратні ресурси, що значно пришвидшує виконання обчислень, особливо при роботі з великими обсягами даних або складними моделями. По друге, його інтерфейс є простим, зрозумілим і не містить нічого зайвого. По-третє, Google Collab забезпечує зручну інтеграцію з Google Drive, що спрощує обробку даних і зберігання результатів роботи.

Для початку, встановлюються всі необхідні бібліотеки, зокрема SuperPoint для виявлення ключових точок, OpenCV для обробки зображень та PyTorch для роботи з моделями глибокого навчання. Наступним кроком підключається і здійснюється вхід у Google Drive. Потім завантажується попередньо навчена модель YOLOv5s для виявлення об'єктів на зображенні.



```python
# Імпорт бібліотек
!pip install superpoint_superglue_deployment opencv-python torch
import cv2
import numpy as np
import torch
from google.colab import drive
from superpoint_superglue_deployment import Matcher
import time
import psutil
import os

# Підключення Google Drive
drive.mount('/content/drive')

# Завантаження YOLO
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
```

Рисунок 3.4 – Імпорт бібліотек

Система перевіряється на наявність графічного процесора, з використанням якого програма працює в рази ефективніше. Хоча Google Colab надає доступ до його використання, але за умов запуску технології в іншому середовищі, на пристрої без GPU, буде використовуватись CPU. Для демонстрації ресурсоємкості системи впроваджені функції, які показують використання CPU, GPU (за умови її наявності), RAM.

```python
# Перевірка наявності GPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Використовується: {device}")

# Функція для отримання використання CPU
def get_cpu_usage():
    return psutil.cpu_percent()

# Функція для отримання використання GPU
def get_gpu_usage():
    if torch.cuda.is_available():
        return torch.cuda.memory_allocated() / (1024 ** 3)  # в ГБ
    return 0

# Функція для обчислення використання пам'яті
def get_memory_usage():
    process = psutil.Process(os.getpid())
    memory_info = process.memory_info()
    return memory_info.rss / (1024 * 1024)  # в МБ
```

Рисунок 3.5 – Перевірка ресурсоємкості

Далі функція load_image завантажує зображення з заданого шляху. Якщо зображення не знайдено, піднімається вийняток FileNotFoundError.

```python
# Функція для завантаження зображення
def load_image(path):
    img = cv2.imread(path)
    if img is None:
        raise FileNotFoundError(f"Зображення {path} не знайдено!")
    return img
```

Рисунок 3.6 – Функція завантаження зображення

Функція detect_objects використовує модель YOLO для виявлення об'єктів на зображенні. Результати зберігаються в detections, що містять координати об'єктів та їхні класи.

```python
# Функція для виявлення об'єктів
def detect_objects(image):
    results = model(image)
    detections = results.xyxy[0].cpu().numpy()
    return detections
```

Рисунок 3.7 – Функція для виявлення об’єктів YOLO

Функція crop_image відповідає за обрізання зображення навколо знайдених об'єктів на поточному зображенні. Вона розраховує нові координати для обрізки з урахуванням співвідношення сторін, що важливо для подальшого порівняння з шаблоном.

```python
# Функція для обрізання зображення
def crop_image(image, x1, y1, x2, y2, aspect_ratio):
    h, w = image.shape[:2]
    obj_width = x2 - x1
    obj_height = y2 - y1
    center_x, center_y = (x1 + x2) // 2, (y1 + y2) // 2

    if obj_width / obj_height > aspect_ratio:
        new_height = int(obj_width / aspect_ratio)
        y1_new, y2_new = max(0, center_y - new_height // 2), min(h, center_y + new_height // 2)
        x1_new, x2_new = x1, x2
    else:
        new_width = int(obj_height * aspect_ratio)
        x1_new, x2_new = max(0, center_x - new_width // 2), min(w, center_x + new_width // 2)
        y1_new, y2_new = y1, y2

    return image[y1_new:y2_new, x1_new:x2_new]
```

Рисунок 3.8 – Функція для обрізання об'єктів з зображення

Функція draw_matches малює лінії між співпадаючими ключовими точками шаблону і поточного зображення. Це допомагає візуально оцінити, наскільки правильно модель знаходить об'єкт зі зразкового зображення.

```python
# Функція для малювання співпадінь між зображеннями
def draw_matches(tmpl_img, curr_img, tmpl_kpts, curr_kpts, matches):
    valid_matches = [
        m for m in matches if m.trainIdx < len(curr_kpts) and m.queryIdx < len(tmpl_kpts)
    ]
    return cv2.drawMatches(
        tmpl_img, tmpl_kpts, curr_img, curr_kpts, valid_matches, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS
    )
```

Рисунок 3.9 – Функція для малювання співпадінь між зображеннями

Функція main відповідає за основну логіку програми.

Спочатку вона завантажує шаблон і поточне зображення, виявляє об'єкти на поточному зображенні, а також обчислює співвідношення сторін для їх обрізки. При відсутності об’єктів, виводиться повідомлення про це і програма припиняє роботу.

```python
# Основна функція
def main(template_path, current_path):
    start_time = time.time()

    tmpl_img = load_image(template_path)
    curr_img = load_image(current_path)

    detections = detect_objects(curr_img)
    if len(detections) == 0:
        print("Не знайдено жодного об'єкта.")
        return

    aspect_ratio = tmpl_img.shape[1] / tmpl_img.shape[0]
```

Рисунок 3.10 – Початок основної функції

Потім налаштовується SuperPoint та SuperGlue для ефективного виявлення ключових точок та обчислення відповідностей між шаблоном і обрізаними зображеннями. Задаються змінні для пошуку об’єкту з найбільшою кількістю співданінь.

```python
# Завантаження SuperPoint та SuperGlue
matcher = Matcher(
    {
        "superpoint": {"input_shape": (-1, -1), "keypoint_threshold": 0.11},
        "superglue": {"match_threshold": 0.3},
        "use_gpu": torch.cuda.is_available(),
    }
)

best_matches = []
best_cropped_img = None
best_tmpl_kpts = []
best_curr_kpts = []
```

Рисунок 3.11 – Налаштування SuperPoint та SuperGlue

Далі проводиться цикл по виявленим об’єктам.

- Виводяться координати кожного об'єкта, впевненість у виявленні та клас.
- Об’єкти на зображенні обрізаються і змінюють розмір до розмірів шаблону.
- Зображення перетворюються в відтінки сірого для подальшої обробки ключових точок і пошуку відповідностей.

```python
for i, det in enumerate(detections):
    if len(det) < 6:
        print(f"Неповні дані для об'єкта {i+1}.")
        continue

    x1, y1, x2, y2 = map(int, det[:4])
    conf, cls = det[4], det[5]

    print(f"\nОб'єкт {i+1}: координати ({x1}, {y1}), ({x2}, {y2}), впевненість: {conf}, клас: {cls}")

    # Обрізання зображення та порівняння
    cropped_img = crop_image(curr_img, x1, y1, x2, y2, aspect_ratio)
    cropped_img = cv2.resize(cropped_img, (tmpl_img.shape[1], tmpl_img.shape[0]))

    tmpl_gray = cv2.cvtColor(tmpl_img, cv2.COLOR_BGR2GRAY)
    cropped_gray = cv2.cvtColor(cropped_img, cv2.COLOR_BGR2GRAY)

    tmpl_kpts, curr_kpts, _, _, matches = matcher.match(tmpl_gray, cropped_gray)
```

Рисунок 3.12 – Обробка знайдених об’єктів

Гомографія допомагає визначити масштабування, обертання, зсув і перспективні зміни об'єкту і необхідне для того, щоб об’єкти на одному зображенні точно відповідали об’єктам на іншому. Виводиться кількість співпадінь кожного об’єкту з шаблоном і потрібним визначається той, у якого збігів найбільше. З ним зберігається зображення, яке демонструє співпадіння.

```
        if len(matches) >= 4:
            print(f"Кількість співпадінь для об'єкту {i+1}: {len(matches)}")

            if len(matches) > len(best_matches):
                best_matches = matches
                best_cropped_img = cropped_img
                best_tmpl_kpts = tmpl_kpts
                best_curr_kpts = curr_kpts

    # Збереження зображення з найкращим співпадінням
    if best_cropped_img is not None and len(best_matches) > 0:
        matched_img = draw_matches(tmpl_img, best_cropped_img, best_tmpl_kpts, best_curr_kpts, best_matches)
        matched_img_path = '/content/drive/My Drive/Photo/matched_image.jpg'
        cv2.imwrite(matched_img_path, matched_img)
        print(f"\nЗображення з найкращим співпадінням збережено: {matched_img_path}")
    else:
        print("Не знайдено жодного об'єкта з достатньою кількістю співпадінь.")
```

Рисунок 3.13 – Гомографія та вибір найкращого об’єкта

В кінці коду викликаються функції, які демонструють ресурсоємкість системи, основна функція для роботи програми, а також задаються шляхи до зображень.

```
    # Виведення часу виконання, використання RAM, CPU та GPU
    end_time = time.time()
    execution_time = end_time - start_time
    memory_usage = get_memory_usage()

    print(f"\nЧас виконання: {execution_time:.2f} секунд")
    print(f"Використано RAM: {memory_usage:.2f} MB")
    print(f"Використання CPU: {get_cpu_usage()}%")
    print(f"Використання GPU: {get_gpu_usage()} GB")


# Виклик основної функції
template_path = '/content/drive/My Drive/Photo/template.jpg'
current_path = '/content/drive/My Drive/Photo/current.jpg'
main(template_path, current_path)
```

Рисунок 3.14 – Шляхи до зображень та виклик функцій

3.3. Результати роботи

Для оцінки ефективності розробленої системи проведено тестування за різними зразками зображень, результати якого демонструють кожен етап роботи програми.

Для першого тесту був обраний випадок з однією ворожою технікою на шаблонному і поточному кадрах. Проте зображення відрізняються масштабом об’єкту, перспективою та освітленням.

Рисунок 3.15 – Зображення для першого тестування

Після запуску програми та підвантаження всіх необхідних бібліотек відпрацьовує детектор YOLO. Був виявлений об’єкт 1 та отримані його координати, впевненість і клас:

```
Об'єкт 1: координати (743, 242), (1176, 698), впевненість: 0.7179301977157593, клас: 8.0
```

Рисунок 3.16 – Результати YOLO на першому тесті

Об’єкт 1 був обрізаний з поточного кадру та підігнаний під роздільну здатність та формат шаблону для оптимізації подальшої роботи шляхом прибирання зайвого фону.

Далі за допомогою SuperPoint і SuperGlue здійснився пошук та порівняння ключових точок об’єкту на шаблоні та обрізаному зображенні поточного кадру. Після гомографії визначено кількість співпадінь та збережено зображення, яке їх демонструє.

```
Кількість співпадінь для об'єкту 1: 60

Зображення з найкращим співпадінням збережено: /content/drive/My Drive/Photo/matched_image.jpg
```

Рисунок 3.17 – Результати SuperPoint, SuperGlue на першому тесті

Рисунок 3.18 – Зображення SuperPoint, SuperGlue для першого тесту

В кінці роботи програми можна побачити, що під час першого тестування при використанні GPU система відпрацювала з такою продуктивністю:

```
Час виконання: 0.68 секунд
Використано RAM: 1425.42 MB
Використання CPU: 30.1%
Використання GPU: 0.2175273895263672 GB
```

Рисунок 3.19 – Продуктивність з GPU для першого тесту

При використанні CPU отримані такі результати:

```
Час виконання: 30.97 секунд
Використано RAM: 935.59 MB
Використання CPU: 30.2%
Використання GPU: 0 GB
```

Рисунок 3.20 – Продуктивність з CPU для першого тесту

Для другого тестування було обране поточне зображення з декількома об'єктами на ньому. Це має показати наскільки правильно система обере ворожу техніку, що відповідає саме шаблону.

Рисунок 3.21 – Зображення для другого тестування

В результаті, після роботи програми, отримані такі дані від YOLO та SuperPoint, SuperGlue:

```
Об'єкт 1: координати (1420, 722), (1761, 969), впевненість: 0.823314368724823, клас: 8.0
Кількість співпадінь для об'єкту 1: 26

Об'єкт 2: координати (1061, 292), (1383, 525), впевненість: 0.6736403107643127, клас: 8.0
Кількість співпадінь для об'єкту 2: 12

Об'єкт 3: координати (790, 539), (1123, 804), впевненість: 0.5171555876731873, клас: 6.0
Кількість співпадінь для об'єкту 3: 14
```

Рисунок 3.22 – Результати детектору для другого тестування

YOLO відпрацював точно й правильно, знайшовши на поточному фото три об'єкти ворожої військової техніки. SuperPoint, SuperGlue, в свою чергу, знайшли співпадіння між цими об'єктами та шаблоном. Потрібною виявилася техніка під номером 1, де співпадінь найбільше. Збережено зображення з результатами роботи.

```
Зображення з найкращим співпадінням збережено: /content/drive/My Drive/Photo/matched_image.jpg
```

Рисунок 3.23 – Виявлення потрібного об’єкту для другого тестування

На отриманому зображенні зрозуміло, що програма правильно виявила об'єкт ворожої техніки, який збігається з машиною на шаблоні.

Рисунок 3.24 – Зображення SuperPoint, SuperGlue для другого тесту

Під час другого тестування при використанні GPU система відпрацювала з такою продуктивністю:

```
Час виконання: 1.48 секунд
Використано RAM: 1427.38 MB
Використання CPU: 64.7%
Використання GPU: 0.24274206161499023 GB
```

Рисунок 3.25 – Продуктивність з GPU для другого тесту

При використанні CPU система дала такий результат:

```
Час виконання: 123.25 секунд
Використано RAM: 936.24 MB
Використання CPU: 57.6%
Використання GPU: 0 GB
```

Рисунок 3.26 – Продуктивність з CPU для другого тесту

Загалом було проведено тестування для вибірки з 6 пар зображень різної складності. В результаті підтверджено, що ефективність системи є на достатньо високому рівні для такого об'єму роботи. У кожному випадку програма правильно виявляла необхідну техніку, а час роботи складав в середньому 1.5 секунди при тестуванні на GPU і 1.5 хвилини на CPU.

Рисунок 3.27 – Додаткові тестування

ВИСНОВКИ

Результатом виконання даної кваліфікаційної роботи є створення інформаційної технології, що представляє собою систему для виявлення й розпізнавання об’єкту ворожої техніки на місцевості за її зразком, і має функціональні можливості для подальшого вдосконалення. Розглянуті класичні й сучасні алгоритми комп’ютерного зору. Досліджено їх основну структуру, принципи й етапи роботи. Методи порівняні та на основі цих даних обрані інструменти для виконання задачі.

У ході виконання кваліфікаційної роботи було виконано такі завдання:

1. Проведено аналіз предметної області для визначення специфічних вимог до системи, яка має забезпечити детектування об’єктів на місцевості за зразками зображення. Це включало огляд технологій алгоритмів комп’ютерного зору, які мають потенціал для ефективного вирішення поставлених завдань.
2. Було визначено логіку роботи програми для детектування та трекінгу об’єктів. Вона включає використання нейронної мережі для виявлення об'єктів YOLO та поєднання SuperPoint і SuperGlue для порівняння об’єктів з шаблоном.
3. Написано програмний код для реалізації системи. Мова Python обрана завдяки її простоті та підтримці великої кількості бібліотек для роботи з алгоритмами комп’ютерного зору. Середовищем виконання обрано Google Colab завдяки доступу до потужних обчислювальних ресурсів та роботі у хмарі.
4. Систему протестовано зображеннями різних рівнів складності. Програма добре показала себе у всіх випадках, забезпечивши високу точність та швидкість виявлення об’єкту на місцевості за його зразком.

Надалі планується:

1. Поглиблена оптимізація коду для забезпечення стабільної роботи програми на обмеженому обладнанні.
2. Дослідження можливості інтеграції більш сучасних методів і версій систем комп’ютерного зору для підвищення точності у надскладних умовах.
3. Інтеграція системи в середовище дрону для реальних випробувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sarlin P.-E., Detone D., Malisiewicz T., Rabinovich A. SuperGlue: Learning Feature Matching with Graph Neural Networks // *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

2. Shi J., Tomasi C. Good Features to Track // *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

3. Radenovic F., Tolias G., Chum O. "Fine-Tuning CNN Image Retrieval with No Human Annotation" // IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018.

4. Москаленко В. В. Моделі і методи інтелектуального аналізу багатовимірних даних за умов апріорної невизначеності : монографія / В. В. Москаленко. – Суми : Сумський державний університет, 2020. – 146 с.

5. Szeliski, R. Computer Vision: Algorithms and Applications., 2010.

6. DeTone, D., Malisiewicz, T., & Rabinovich, A. SuperPoint: Self-Supervised Interest Point Detection and Description. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018.

7. Ruben Nogales, Marco E. Benalcázar Analysis and Evaluation of Feature Selection and Feature Extraction Methods, 2023.

8. Lowe, David G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 2004.

9. Застосування методу SURF у системах контролю та управління доступом на основі біометричних технологій [Електронний ресурс] URL: https://habr.com/ru/articles/152679/

10. Hao Li, Lipo Wang, Tianyun Zhao, Wei Zhao Local-Peak Scale-Invariant Feature Transform for Fast and Random Image Stitching, 2024.

11. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

12. Redmon, Joseph You only look once: Unified, real-time object detection. *// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

13. Sarlin, P.-E., DeTone, D., Malisiewicz, T., & Rabinovich, A. SuperGlue: Learning Feature Matching with Graph Neural Networks. *// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

14. Dusser, A., Tokmakov, V., & Fua, P. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features *// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

15. Кук, К. Л. Множник тихої сили: історія та роль БПЛА у війні. *// IEEE Aerospace Conference, 1-7*, 2007.

16. Саттон, Г. І. Неймовірний успіх українського Bayraktar TB2: Привид острова Зміїний. Naval News, 2022.

17. Петтіджон, Стейсі. Еволюція, а не революція: війна безпілотників під час вторгнення Росії в Україну 2022 року. Центр нової американської безпеки, 2024.

18. Як працює Object Tracking на YOLO [Електронний ресурс] URL: https://habr.com/ru/articles/514450/

19. Tomasz Malisiewicz SuperPoint and SuperGlue: Lessons Learned, 2022.

20. Rocco, I., Cimpoi, M., Maji, S., Arandjelović, R., Torii, A., Pajdla, T., & Sivic, J. End-to-End Weakly-Supervised Semantic Alignment. *// In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

21. Bai, S., & Urtasun, R. SINKHORN: A Review of Regularization Techniques for Bipartite Matching. Neural Information Processing Systems, 2020.

ДОДАТОК А

```
# Імпорт бібліотек
!pip install superpoint_superglue_deployment opencv-python torch
import cv2
import numpy as np
import torch
from google.colab import drive
from superpoint_superglue_deployment import Matcher
import time
import psutil
import os

# Підключення Google Drive
drive.mount('/content/drive')

# Завантаження YOLO
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

# Перевірка наявності GPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Використовується: {device}")

# Функція для отримання використання CPU
def get_cpu_usage():
    return psutil.cpu_percent()

# Функція для отримання використання GPU
def get_gpu_usage():
```

```
    if torch.cuda.is_available():
        return torch.cuda.memory_allocated() / (1024 ** 3)  # в ГБ
    return 0

# Функція для обчислення використання пам'яті
def get_memory_usage():
    process = psutil.Process(os.getpid())
    memory_info = process.memory_info()
    return memory_info.rss / (1024 * 1024)  # в МБ

# Функція для завантаження зображення
def load_image(path):
    img = cv2.imread(path)
    if img is None:
        raise FileNotFoundError(f"Зображення {path} не знайдено!")
    return img

# Функція для виявлення об'єктів
def detect_objects(image):
    results = model(image)
    detections = results.xyxy[0].cpu().numpy()
    return detections

# Функція для обрізання зображення
def crop_image(image, x1, y1, x2, y2, aspect_ratio):
    h, w = image.shape[:2]
    obj_width = x2 - x1
    obj_height = y2 - y1
    center_x, center_y = (x1 + x2) // 2, (y1 + y2) // 2
```

```
            if obj_width / obj_height > aspect_ratio:
                new_height = int(obj_width / aspect_ratio)
                y1_new, y2_new = max(0, center_y - new_height // 2), min(h, center_y
+ new_height // 2)
                x1_new, x2_new = x1, x2
            else:
                new_width = int(obj_height * aspect_ratio)
                x1_new, x2_new = max(0, center_x - new_width // 2), min(w, center_x
+ new_width // 2)
                y1_new, y2_new = y1, y2

            return image[y1_new:y2_new, x1_new:x2_new]

        # Функція для малювання співпадінь між зображеннями
        def draw_matches(tmpl_img, curr_img, tmpl_kpts, curr_kpts, matches):
            valid_matches = [
                m for m in matches if m.trainIdx < len(curr_kpts) and m.queryIdx <
len(tmpl_kpts)
            ]
            return cv2.drawMatches(
                    tmpl_img, tmpl_kpts, curr_img, curr_kpts, valid_matches, None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS
            )

        # Основна функція
        def main(template_path, current_path):
            start_time = time.time()
```

```
tmpl_img = load_image(template_path)
curr_img = load_image(current_path)

detections = detect_objects(curr_img)
if len(detections) == 0:
    print("Не знайдено жодного об'єкта.")
    return

aspect_ratio = tmpl_img.shape[1] / tmpl_img.shape[0]

# Завантаження SuperPoint та SuperGlue
matcher = Matcher(
    {
        "superpoint": {"input_shape": (-1, -1), "keypoint_threshold": 0.008},
        "superglue": {"match_threshold": 0.4},
        "use_gpu": torch.cuda.is_available(),
    }
)

best_matches = []
best_cropped_img = None
best_tmpl_kpts = []
best_curr_kpts = []

for i, det in enumerate(detections):
    if len(det) < 6:
        print(f"Неповні дані для об'єкта {i+1}.")
        continue
```

```
            x1, y1, x2, y2 = map(int, det[:4])
            conf, cls = det[4], det[5]

                print(f"\nОб'єкт {i+1}: координати ({x1}, {y1}), ({x2}, {y2}), впевненість: {conf}, клас: {cls}")

            # Обрізання зображення та порівняння
            cropped_img = crop_image(curr_img, x1, y1, x2, y2, aspect_ratio)
                    cropped_img = cv2.resize(cropped_img, (tmpl_img.shape[1], tmpl_img.shape[0]))

            tmpl_gray = cv2.cvtColor(tmpl_img, cv2.COLOR_BGR2GRAY)
           cropped_gray = cv2.cvtColor(cropped_img, cv2.COLOR_BGR2GRAY)

                tmpl_kpts, curr_kpts, _, _, matches = matcher.match(tmpl_gray, cropped_gray)

            if len(matches) >= 4:
               print(f"Кількість співпадінь для об'єкту {i+1}: {len(matches)}")

               if len(matches) > len(best_matches):
                  best_matches = matches
                  best_cropped_img = cropped_img
                  best_tmpl_kpts = tmpl_kpts
                  best_curr_kpts = curr_kpts

        # Збереження зображення з найкращим співпадінням
        if best_cropped_img is not None and len(best_matches) > 0:
```

```
                matched_img = draw_matches(tmpl_img, best_cropped_img, best_tmpl_kpts, best_curr_kpts, best_matches)
                matched_img_path = '/content/drive/My Drive/Photo/matched_image.jpg'
            cv2.imwrite(matched_img_path, matched_img)
                print(f"\nЗображення з найкращим співпадінням збережено: {matched_img_path}")
        else:
            print("Не знайдено жодного об'єкта з достатньою кількістю співпадінь.")

        # Виведення часу виконання, використання RAM, CPU та GPU
        end_time = time.time()
        execution_time = end_time - start_time
        memory_usage = get_memory_usage()

        print(f"\nЧас виконання: {execution_time:.2f} секунд")
        print(f"Використано RAM: {memory_usage:.2f} MB")
        print(f"Використання CPU: {get_cpu_usage()}%")
        print(f"Використання GPU: {get_gpu_usage()} GB")

    # Виклик основної функції
    template_path = '/content/drive/My Drive/Photo/template.jpg'
    current_path = '/content/drive/My Drive/Photo/current.jpg'
    main(template_path, current_path)
```