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

Навчально-науковий Фізико-технічний інститут

Кафедра математичного моделювання та аналізу даних

«До захисту допущено»:

В.о. завідувач кафедри

_____ Іван ТЕРЕЩЕНКО

«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Математичні методи моделювання, розпізнавання образів та безпеки даних»

спеціальності 113 «Прикладна математика»

на тему: «Алгоритм розпізнавання цілі із використанням попередньої фільтрації»

Виконав:

студент IV курсу, групи ФІ-91

Кухар Богдан Вікторович

Керівник:

Старший викладач Тітков Дмитро Валерійович

Рецензент:

Професор каф. АМЕС. к.т.н. Володимир Пілінський

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторі без відповідних посилань

Студент _____

Київ – 2023 року

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

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

Освітня програма «Математичні методи моделювання, розпізнавання образів та безпеки даних»

ЗАТВЕРДЖУЮ

В.о. завідувач кафедри

_____ Іван ТЕРЕЩЕНКО

«__» _____ 2022 р.

ЗАВДАННЯ

на дипломну роботу

Студент : Кухар Богдан Вікторович

1. Тема роботи «Алгоритм розпізнавання цілі із використанням попередньої фільтрації»,

   керівник : ст. викладач Тітков Дмитро Валерійович

   затверджені наказом по університету від «__» ____ 2022р. №_____

2. Термін подання студентом роботи _____.

3. Вхідні дані до роботи – *розвинені методи трекінгу об’єктів*.

4. Зміст роботи – *досліджуючи основні вразливі місця систем спостереження побудовано алгоритм автоматичного відстеження об'єктів за допомогою комбінації відомих трекерів та методів попередньої фільтрації.*

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): *Презентація доповіді*

6. Дата видачі завдання 16.02.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд опублікованих джерел за тематикою дослідження	Січень-лютий 2023 р.	
2	Підготовка першого розділу	Лютий-березень 2023 р.	
3	Підготовка другого розділу	Березень 2023 р.	
4	Підготовка даних до роботи	Квітень 2023 р.	
5	Аналіз даних	Квітень-травень 2023 р.	
6	Підготовка третього розділу	Травень 2023 р.	
7	Оформлення дипломної роботи	Травень 2023 р.	
8	Підготовка презентації доповіді	Травень 2023 р.	

Студент _____ Кухар Б.В.

Керівник _____ Тітков Д.В.

РЕФЕРАТ

Кваліфікаційна робота містить : 48 стор., 18 рисунків, 2 таблиці та 21 джерел.

Мета роботи полягає у вдосконаленні наявних алгоритмів відслідковування об’єктів шляхом попередньої фільтрації зображення. Об’єктом дослідження є сучасні системи спостереження. Тоді алгоритми трекінгу цілей та методи попередньої фільтрації зображення – предмет дослідження.

У роботі було опрацьовано проблеми сучасних систем спостереження, що дало можливість коректно поставити та реалізувати задачу ефективного алгоритму захоплення цілі без безпосередньої участі оператора у реальному часі. В його основі лежать метод трекінгу MIL Tracker та спосіб попередньої фільтрації за допомогою Лапласіана. Результати дослідження наведені у вигляді порівняльних таблиць та знімків роботи покращеного алгоритму на різних відеопотоках.

ТРЕКІНГ, ФІЛЬТРАЦІЯ, КОРЕЛЯЦІЯ, ПОРОГОВА СЕГМЕНТАЦІЯ, ЛАПЛАСІАН

ABSTRACT

The qualifying work consists of: 48 pages, 18 figures, 2 tables, and 21 references.

The purpose of the work is to enhance existing object tracking algorithms through pre-filtering of images. The research focuses on modern surveillance systems, where target tracking algorithms and image pre-filtering methods are the subjects of investigation.

The work addresses the issues of modern surveillance systems, enabling the proper formulation and implementation of the task of efficient target acquisition algorithm without direct operator involvement in real-time. The MIL Tracker tracking method and pre-filtering using Laplacian filtering are the foundation of the proposed algorithm. The results of the work are presented in the form of comparative tables and snapshots of the improved algorithm's performance on various video streams.

TRACKING, FILTERING, CORRELATION, THRESHOLD SEGMENTATION, LAPLACIAN

ЗМІСТ

Вступ

Застосування алгоритмів трекінгу в системах спостереження є однією з важливих технологій, яка забезпечує високу ефективність і точність відстеження об'єктів на відеозаписах. Спостереження є актуальною проблемою в сучасному світі, оскільки воно використовується для безпеки, контролю і виявлення подій в різних сферах, таких як дорожній рух, відеоспостереження та розпізнавання облич. Однією з найактуальніших для України галузей де використовується трекінг є військова справа.

Завдяки розвитку технологій безпілотних літальних апаратів (дронів), військові сили мають можливість отримувати розширену інформацію з повітряного простору та вести нагляд за територіями, де ведуться бойові дії. Застосування алгоритмів трекінгу у дронах в сфері військових операцій стає все більш актуальним і важливим завданням.

Очікується, що результати цієї дипломної роботи допоможуть удосконалити системи трекінгу у дронах для військових операцій, забезпечити високу точність, надійність та швидкодію відстеження об'єктів. Такі вдосконалення можуть мати велике значення для підвищення ефективності військових операцій та безпеки воїнів.

Актуальність роботи :

Забезпечення точного та надійного трекінгу об'єктів у реальному часі є ключовим аспектом при використанні дронів у військових операціях. Алгоритми трекінгу дозволяють дронам автоматично виявляти, відстежувати та залишатися сфокусованими на цілях у руху, що забезпечує важливу підтримку для збройних сил на полі бою.

У роботі пропонуються методи покращення алгоритмів трекінгу за допомогою попередньої фільтрації відеопотоку, це дозволить виконувати захоплення і супровід цілі ефективніше.

Мета і завдання:

Метою дослідження є вдосконалення наявних алгоритмів відслідковування об’єктів для сучасних систем спостереження шляхом попередньої фільтрації зображення.

Завдання роботи полягає у дослідженні роботи алгоритмів трекінгу об’єктів, виявленні їхніх слабких та сильних сторін без попередньої фільтрації зображення та разом з нею.

Об’єкт дослідження: сучасні системи спостереження та методи відслідковування цілей.

Предмет дослідження : алгоритми трекінгу цілей та методи попередньої фільтрації зображення

Методи дослідження :

- Методи просторової фільтрації зображення
- Методи підвищення різкості за допомогою похідних
- Методи корекції яскравісної складової зображення
- Алгоритми трекінгу (MOSSE, MIL,BOOSTING,MEDIANFLOW,TLD,KCF)

Наукова новизна у дослідженні :

Дослідження містить розробки, що удосконалюють наявні алгоритми трекінгу за допомогою попередньої фільтрації. Цей підхід дозволяє системі-трекеру краще та ефективніше супроводжувати об’єкти у реальному часі.

2 СИСТЕМИ СПОСТЕРЕЖЕННЯ FPV ДРОНІВ У ХОДІ БОЙОВИХ ДІЙ

2.1 Принцип роботи FPV дронів у ході бойових дій

FPV (First Person View) дрон - це безпілотний літальний апарат (дрон), оснащений відеокамерою та передавачем, який передає відеозображення з камери на приймач, що знаходиться у відеогарнітурі або на моніторі пілота.

Системи автоматичного трекінгу об'єктів за допомогою FPV дронів в ході бойових дій використовуються для виявлення, слідкування та спостереження за певними об'єктами чи особами на полі бою. Ці системи дозволяють операторам ефективно вести розвідку, контролювати рухи ворожих сил та забезпечувати підтримку з повітря.

Ось деякі основні компоненти та характеристики систем автоматичного трекінгу FPV дронів у ході бойових дій:

1. *Високоефективна відеокамера:* Для автоматичного трекінгу потрібна високоякісна відеокамера з високою роздільною здатністю та швидким оновленням зображення. Це дозволяє точно виявляти та відстежувати об'єкти, навіть при швидких рухах.
2. *Алгоритми комп'ютерного зору:* Системи автоматичного трекінгу використовують складні алгоритми комп'ютерного зору для виявлення об'єктів на зображенні та визначення їх положення. Це може включати виявлення форми, розміру, руху, колірних ознак тощо.
3. *Автоматична система стабілізації:* Для точного трекінгу об'єктів під час польоту FPV дрон повинен мати систему стабілізації, яка компенсує вібрації та рухи, забезпечуючи стабільне зображення для аналізу та трекінгу.
4. *Система передачі даних:* Для ефективної роботи системи трекінгу, відеодані з FPV дрона передаються в реальному часі на приймач. Це може

використовувати бездротові технології передачі даних, такі як Wi-Fi або RF-системи.

5. *Інтеграція зі земною командною станцією*: Система трекінгу FPV дрону повинна бути інтегрована зі земною командною станцією, де оператор може аналізувати та контролювати процес трекінгу. Це може включати відображення відео, інтерфейс для вибору цілей, керування дроном тощо.

2.2 Основні проблеми, що виникають під час спостереження

Під час автоматичного трекінгу об'єктів за допомогою FPV дронів у ході бойових дій можуть виникати різні проблеми, які можуть вплинути на ефективність та результативність системи. Деякі з цих проблем включають:

1. *Відмова або нестабільна робота системи трекінгу*: Технологія автоматичного трекінгу може зазнавати помилок або несправностей, що можуть призвести до втрати відстеження об'єктів. Це може статися через проблеми з алгоритмами комп'ютерного зору, втрату сигналу передачі даних або проблеми зі стабілізацією зображення.
2. *Перешкоди на полі бою*: Наявність фізичних перешкод, таких як будівлі, дерева, гірські формації тощо, може заважати точному трекінгу об'єктів. Це може призвести до втрати об'єкта зору дрона або помилкового трекінгу неправильного об'єкта.
3. *Відсутність чітких ознак для трекінгу*: Якщо об'єкти на полі бою не мають визначених чітких ознак або рухаються непередбачувано, може бути важко їх точно виявити та відстежувати. Це може статися, наприклад, у випадку маскування, зміни зовнішнього вигляду або швидких змін траєкторії руху об'єкта.
4. *Втрата відео каналу або ж каналу керування*: У ході бойових дій може бути велика кількість джерел електромагнітних перешкод, таких як радіопередавачі, радари, імпульсні перешкоджувачі, комунікаційні системи

тощо. Ці сигнали можуть впливати на передачу сигналу керування FPV дроном і спричиняти його втрату.

Отже, основою ідеєю роботи є розробка такого ефективного алгоритму для захоплення цілі, який б обходив усі вище описані проблеми та міг продовжити супровід цілі без безпосередньої участі оператора у разі втрати каналів зв’язку з дроном.

3 ОСНОВНІ МЕТОДИ ПОПЕРЕДНЬОЇ ФІЛЬТРАЦІЇ ЗОБРАЖЕННЯ ДЛЯ УДОСКОНАЛЕННЯ ПРОЦЕСУ ЗАХОПЛЕННЯ ЦІЛІ

На практиці вихідні зображення зазвичай не мають високої якості через наявність помилок і шумів під час опрацювання даних. Для того щоб покращити якість розпізнавання треба застосувати один із методів попередньої фільтрації. Зокрема, буде використана просторова фільтрація.

Просторова фільтрація зображень включає такі кроки:

1.Визначення центральної точки (x, y) околу;

2.Виконання операції з використанням значень пікселів з певної області, яка оточує центральну точку;

3.Призначення результату цієї операції центральній точці;

4.Повторення процесу для кожної точки зображення. При цьому зміна позиції центральної точки утворює нові області, що відповідають кожному пікселю зображення.

Іншими словами, просторова фільтрація зображень є процесом виконання згортки зображення з ядром фільтру. Лінійна фільтрація зображень в просторовій області полягає у визначенні лінійної комбінації значень яскравості пікселів у вікні фільтрації з використанням коефіцієнтів матриці ваг, яку називають маскою або ядром лінійного фільтру.

3.1 Медіанна фільтрація

Медіанні фільтри корисні для зменшення випадкового шуму, особливо коли щільність ймовірності амплітуди шуму має великі хвости та періодичні моделі. Медіанний процес фільтрації виконується ковзанням вікна над зображенням.

Медіанний фільтр по черзі розглядає кожен піксель на зображенні та дивиться на його найближчих сусідів, щоб вирішити, чи є він репрезентативним для свого оточення. Замість простої заміни піксельного значення на середнє значення сусідніх піксельних значень, воно замінює його медіаною цих значень. Медіана обчислюється шляхом сортування всіх значень пікселів із навколишнього середовища в числовому порядку, а потім заміни пікселя, який розглядається, середнім значенням пікселя. (Якщо околиця, що розглядається, містить парну кількість пікселів, використовується середнє значення двох середніх пікселів.)

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Сусідні значення :
115,119,120,123,124
125,126,127,150

Медіана : 124

Рисунок 3.1 - Приклад розрахунку медіани на частині зображення

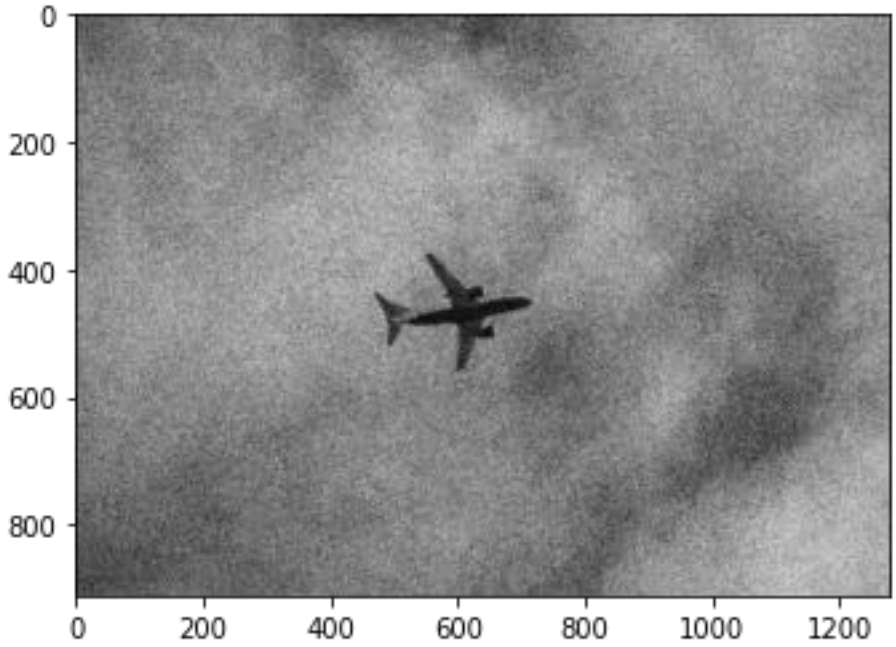

Рисунок 3.2 - Приклад зображення з шумами

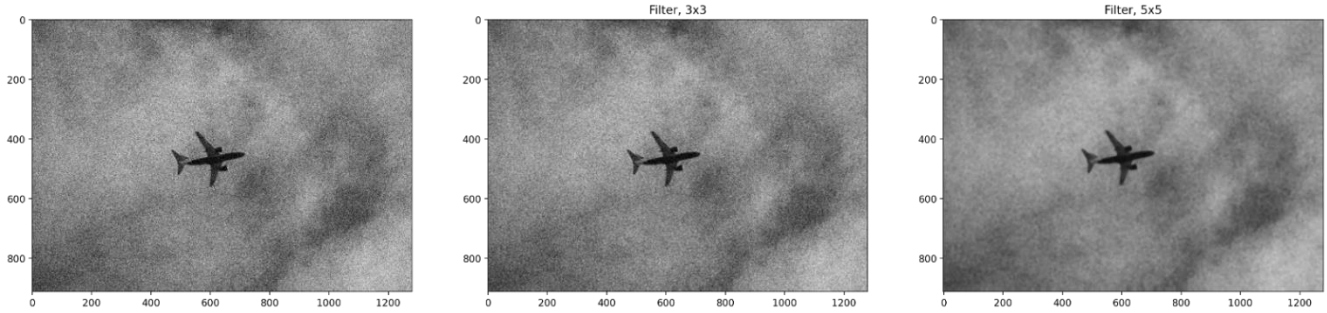

Рисунок 3.3 - Застосування медіанної фільтрації

3.2 Порогова обробка(Threshold segmentation)

Порогова сегментація зображення за рівнями яскравості - найпростіший вид сегментації зображення. Цей метод заснований на тому, що багато об'єктів або області зображення характеризуються постійною відбивною здатністю або поглинанням світла на їх поверхні. Відмінною рисою порогової сегментації є обчислювальна ефективність і можливість використання в системах реального масштабу часу.

Бінаризація є процесом перетворення зображення з пороговою характеристикою на бінарне зображення, де кожен піксель може мати значення лише 0 або 1. Цей процес може бути корисним у випадку, коли важливо виділити контури об'єктів на зображенні, а деталі всередині об'єктів або фону не є істотними. Однак визначення порогового значення є ключовим етапом цього процесу, оскільки він визначає, які пікселі мають значення 0, а які - 1. Застосування бінаризації дозволяє досягти більшої наочності при візуальному сприйнятті, ніж вхідне зображення з пороговою характеристикою.

Порогова сегментація виконується наступним чином:

$$\mathrm{B}_{\mathrm{in}_{\mathrm{i,j}}} = \begin{cases} 1, I_{i,j} \geq \mathrm{T} \\ 0, I_{i,j} < \mathrm{T} \end{cases} \tag{3.1}$$

де Bini, - елемент результуючого бінарного зображення, Ii,j- елемент вихідного зображення.

Успіх порогової сегментації залежить від способу вибору порогу T. У разі використання методу Оцу, зображення розбивається на передній і на задній план, тобто, проводиться бінаризація зображення.

У рамках виявлення повітряних об’єктів порогова обробка є ефективним інструментом за рахунок контрасту між об’єктом та небом у світлий час доби. Відповідно пікселі об’єкта та фону будуть відходити до різних категорій при бінаризаціїї зображення(чорне та біле) і добре виділятися встановленим порогом.

Оптимальне значення порогу при високій контрастності T=101.

3.3 Гамма-корекція яскравості

Наше зорове сприйняття відрізняється від сприйняття цифрових пристроїв, тому існує необхідність у гамма-корекції зображень. Цей процес

дозволяє привести цифрове кодування зображення у відповідність з нашим зоровим сприйняттям зображення. Гамма-корекція є одним зі статичних перетворень і полягає в зміні яскравості зображення за допомогою відповідного математичного виразу.

$$S_{\text{вих}} = \text{c} \cdot S_{\text{вх}}^{\gamma} \tag{3.2}$$

де c та γ – додатні константи, а sвх і sвих - відповідно значення яскравості на вході та виході процедури її зміни.

Вигляд відповідних співвідношень наведений на зображені :

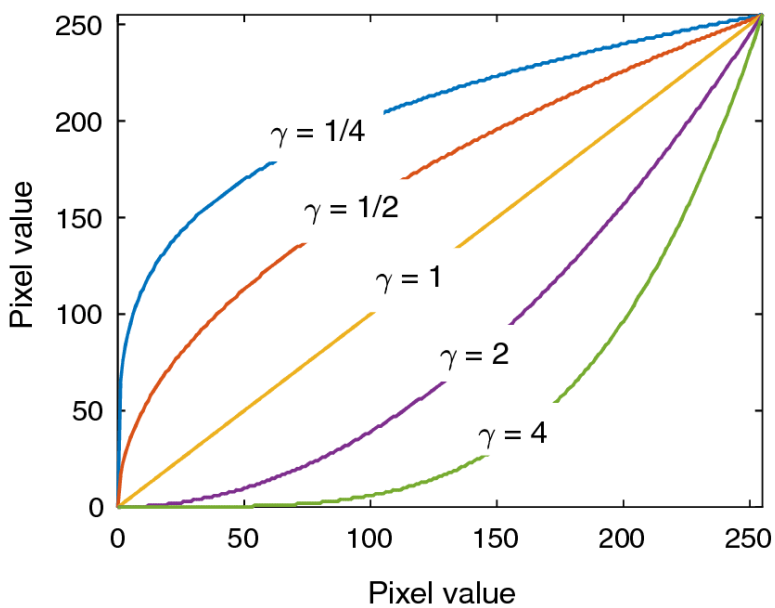

Рисунок 3.4 – Залежність значень вхідного та вихідного пікселю від параметра гамма[17]

3.4 Логарифмічне перетворення

Представляється у вигляді формули :

$$S = c \cdot \log(1 + r) \tag{3.3}$$

де c – константа, що формує інтенсивність перетворення, r – значення яскравості вхідного пікселя.

Форма логарифмічної кривої вказує на те, що це перетворення перетворює вузький діапазон низьких значень яскравості на вихідному зображенні в ширший діапазон значень. На великих значеннях вхідного сигналу відбувається навпаки. Для розтягування діапазону темних пікселів на зображенні з одночасним стисканням діапазону значень яскравих пікселів доцільно використовувати цей тип перетворення. Зворотне логарифмічне перетворення, навпаки, розтягує діапазон яскравих пікселів і стискає діапазон темних пікселів.

Загалом перетворення даного типу корисне саме при виявленні об'єктів у темні періоди доби.

3.5 Підвищення різкості зображення за допомогою похідних

Підвищення різкості зображення можна досягти за допомогою фільтру Лапласа.

$$g(x, y) = \nabla^2 f(x, y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{3.4}$$

Технічно кажучи, ми перевіряємо зміну інтенсивності пікселів, оскільки ми завжди працюємо із зображеннями у градаціях сірого, і найкращий спосіб побачити різні кольори в градаціях сірого – це перевірити інтенсивність пікселів.

Отже, якщо є різкість інтенсивності пікселів, можна сказати, що є перевага. Тепер, щоб виявити край, нам потрібно зробити це математично. Для цього беремо похідну від інтенсивності і знаходимо різкість інтенсивності, де б вона не була.

Фільтри беруть одну похідну та знаходять ребро в одному з вимірів (x або y). Але за допомогою фільтра Лапласа ми можемо отримати краї в обох вимірах, тому ми беремо подвійну похідну від інтенсивностей. І коли виконується подвійне виведення, графік вказує на нуль. Тож перевіряються ті пікселі, які ведуть до нуля, а потім відповідні точки позначаються як крайові.

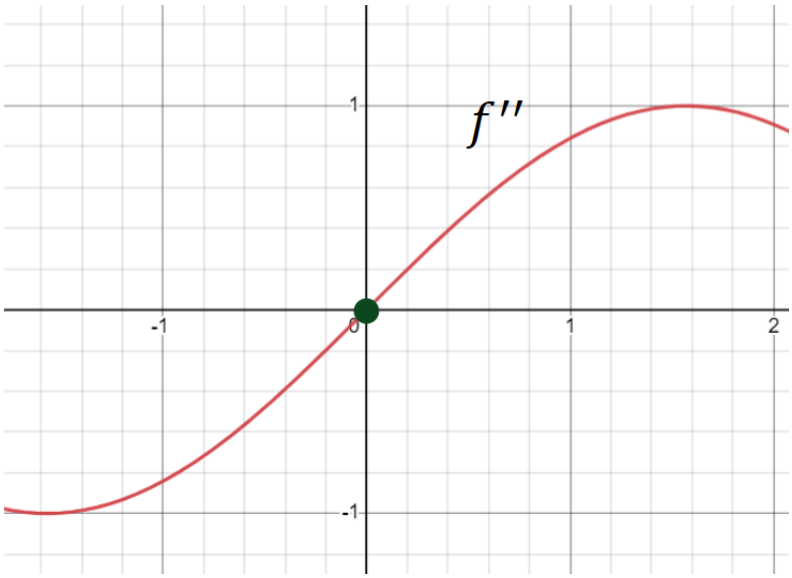

Рисунок 3.5 – Відображення крайової точки за допомогою похідної

Результат застосування даного методу підкреслює різкі зміни яскравостей на зображенні та пригнічує області зі слабкими змінами яскравості. Це призводить до отримання зображення, на якому розриви та контури відображаються яскравіше, але фон залишається темним. Проте можна відновити фон з одночасним збереженням підвищення різкості зображення шляхом віднімання лапласіана від початкового зображення.

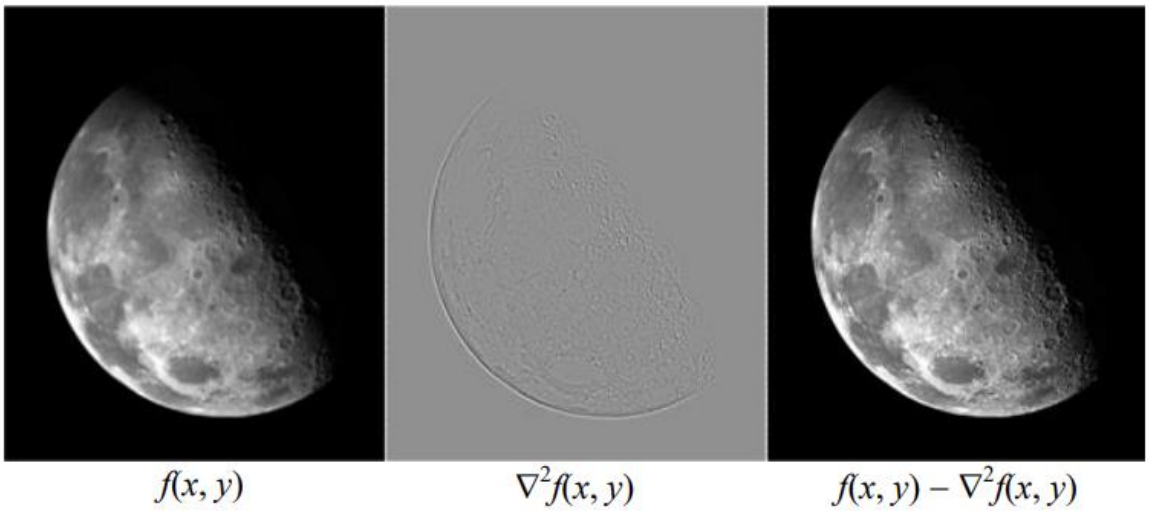

Рисунок 3.6 – Результат роботи фільтру Лапласа та віднімання від оригінального зображення

Для одновимірного аналізу зображення, мова частіше йде про виявлення дефектів по одній з осей. Використовують фільтри, що базуються на першій похідній, яку знаходять через модуль градієнта, що визначається як :

$$\nabla f = |\nabla f| = \sqrt{G_x^2 + G_y^2} = \left[(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2\right]^{\frac{1}{2}} \quad (3.5)$$

4 ЗАДАЧА ТРЕКІНГУ ТА ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ВІДСТЕЖЕННЯ ОБ’ЄКТІВ

Трекінг об'єктів, також відомий як відстеження об'єктів або слідкування за об'єктами, є процесом визначення та прогнозування руху об'єктів на зображенні або відео з моменту їх появи до зникнення або пересування за межі зображення.

Метою трекінгу об'єктів є відстеження траєкторій руху об'єктів у відеопотоці, виявлення змін у їх положенні, формі, розмірі або орієнтації в часі та передбачення їх майбутнього стану. Це може бути корисно для багатьох застосувань, включаючи автоматичне відстеження об'єктів на відеозаписах, відеоспостереженні, аналізі руху, розпізнаванні жестів, віртуальній реальності, машинному зору, автономних транспортних засобах тощо.

4.1 Математична модель трекінгу

Математична модель трекінгу об'єктів включає в себе різні підходи, але в загальному вона складається з декількох основних компонентів:

1. Модель стану об'єкта: Це модель, яка описує поведінку та рух об'єкта, що відстежується. Вона може бути представлена в формі математичних рівнянь, таких як лінійні або нелінійні диференціальні рівняння, або ж у вигляді статистичних моделей, наприклад, калманівських фільтрів.
2. Модель вимірювань: Ця модель описує, як об'єкт взаємодіє із засобами спостереження. Вона може включати параметри, такі як розташування, розмір, швидкість тощо. Модель вимірювань дозволяє отримати оцінку стану об'єкта на основі отриманих вимірювань.
3. Функція вимірювання: Це функція, яка пов'язує реальні вимірювання, отримані зі спостережень, з параметрами моделі стану. Ця функція може бути

детермінованою або стохастичною, залежно від точності та надійності вимірювань.

4. Алгоритм трекінгу: Це алгоритм, який використовує модель стану, модель вимірювань та функцію вимірювання для оцінки та прогнозування стану об'єкта в кожний момент часу. По суті, це алгоритм фільтрації, який використовує отримані вимірювання для коригування та оновлення прогнозів про стан об'єкта.
5. Переслідування об'єкта: Цей етап включає апдейт моделі стану та вимірювань з кожним новим спостереженням та прогнозує майбутній стан об'єкта, зважаючи на динаміку ймовірностей та варіацій у вимірюваннях.

Математична модель трекінгу об'єктів може бути реалізована за допомогою різних методів, таких як фільтрація Калмана, фільтрація частинок, фільтрація оптимальної лінійної оцінки (ОЛО), розширена фільтрація Калмана (EKF), нечітка логіка тощо. Кожен з цих методів має свої переваги та обмеження, і вибір конкретного методу залежить від контексту задачі трекінгу об'єктів.

4.2 Огляд методів відстеження об'єктів

Більшість алгоритмів відстеження об'єктів використовують підхід, відомий як відстеження за виявленням. Цей підхід передбачає використання незалежного детектора, який застосовується до всіх кадрів зображення для виявлення об'єктів, та трекера, який використовує результати детектора. Дані виявлення на кожному кадрі використовуються для відстеження шляхом їх з'єднання та призначення ідентичних ідентифікаторів обмежуючим рамкам, що містять той самий об'єкт.

Залежно від того, як використовуються кадри відео, відстеження об'єктів може працювати за двома методами:

1. Пакетний метод: алгоритми використовують інформацію з майбутніх відеокадрів для визначення ідентичності об'єкта в поточному кадрі. Цей метод надає кращу якість відстеження.

2. Онлайновий метод: алгоритми використовують лише поточну та попередню інформацію для прийняття рішень щодо поточного кадру. Онлайнові методи зазвичай працюють гірше, оскільки вони обмежені лише поточним кадром. Однак цей метод необхідний для роботи в реальному часі.

У трекінгу об'єктів найпоширенішим методом є використання згорткових нейронних мереж (Convolutional Neural Networks, CNN). Проте також існують інші методи, такі як рекурентні нейронні мережі (Recurrent Neural Networks, RNN), автокодери (Autoencoders, AE), генеративні змагальні мережі (Generative Adversarial Networks, GAN), сіамські нейронні мережі (Siamese Neural Networks, SNN) та інші алгоритми, що не базуються на нейронних мережах.

Найпопулярніші реалізації методів відстеження об'єктів включають:

1. Відстеження об'єктів OpenCV: OpenCV надає різні трекери, такі як BOOSTING, MIL, KCF, CSRT, MedianFlow, TLD, MOSSE та GOTURN. Кожен з цих трекерів має свої переваги та недоліки, і вибір залежить від конкретних потреб користувача.

2. Багатодоменна згорткова нейронна мережа (Multi-domain Convolutional Neural Network, MDNet): Цей метод використовує попередньо навчену згорткову мережу на великому наборі відео, щоб отримати загальні представлення об'єктів. При відстеженні об'єктів у новій послідовності створюється нова мережа, яка об'єднує спільні шари з попередньо навченої згорткової мережі. Цей метод показує високу точність відстеження і може працювати у реальному часі.

3. Simple Online and Realtime Tracking (SORT): Це ефективний підхід до відстеження кількох об'єктів в реальному часі. Він використовує просту комбінацію відомих методів, таких як фільтр Калмана, і досягає точності, порівняної з іншими онлайн-трекерами. SORT має високу швидкодію оновлення трекера, що дозволяє працювати зі швидкістю до 260 Гц.

4.3 Опис обраних для подальшого дослідження методів трекінгу об'єктів.

4.3.1 Boosting tracker

Оскільки ми цікавимося відстеженням, ми припускаємо, що цільовий об'єкт вже був виявлений. Ця область зображення вважається позитивним зразком для трекера. В той же час негативні зразки вилучаються шляхом взяття областей такої ж розмірності, як вікно цільового об'єкта, з оточуючого фону. Ці зразки використовуються для кількох ітерацій алгоритму online boosting з метою отримання першої моделі, яка вже є стабільною. Зверніть увагу, що ці ітерації потрібні лише для ініціалізації трекера.

Крок відстеження базується на класичному підході відстеження за шаблоном. Ми оцінюємо поточний класифікатор у регіоні інтересу і отримуємо для кожної позиції значення впевненості. Після цього аналізуємо карту впевненості і зсуваємо вікно цільового об'єкта до нового положення максимуму. Для виявлення максимуму також можна використовувати процедуру зсуву середнього значення . Використання моделі руху для цільового об'єкта дозволяє зменшити область пошуку. Після виявлення об'єкта класифікатор має бути оновлений, щоб

адаптуватися до можливих змін у зовнішньому вигляді цільового об'єкта і стати розрізнювальним щодо різних фонів. Поточний регіон цільового об'єкта використовується як позитивне оновлення класифікатора, тоді як оточуючі регіони представляють негативні зразки. З появою нових кадрів вся процедура повторюється, і класифікатор здатний адаптуватися до можливих змін у вигляді об'єкта і стає стійким до шуму в фоновому середовищі.

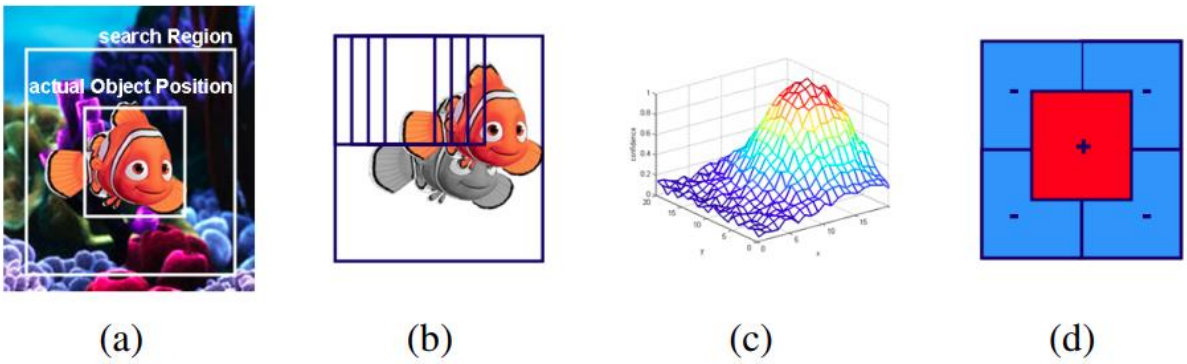

Рисунок 4.1 - Чотири основні кроки відстеження за допомогою класифікатора[18]

Враховуючи початкове положення об'єкта (a) у момент часу t, класифікатор оцінюється у багатьох можливих положеннях у пошуковій області навколо вікна в кадрі t + 1. Отримана карта впевненості (c) аналізується для оцінки найімовірнішого положення, а потім оновлюється трекер (класифікатор) (d)

Boosting Tracker базується на онлайн-версії алгоритму AdaBoost - алгоритм збільшує ваги неправильно класифікованих об'єктів, що дозволяє слабкому класифікатору "фокусуватися" на їх виявленні. Але, до того як зрозуміти справжню суть алгоритму варто ввести декілька допоміжних понять:

1. *Слабкий класифікатор:* Слабкий класифікатор повинен давати результати, трохи кращі за випадкове вгадування (тобто для бінарної задачі прийняття рішення, показник помилок повинен бути менше 50%). Гіпотеза h^{weak}, створена слабким

класифікатором, відповідає певній ознаці і отримується за допомогою визначеного алгоритму навчання.

2. *Селектор:* Заданий набір з M слабких класифікаторів з $H^{weak} = \{h_1^{weak}, \dots, h_m^{weak}\}$, селектор обирає один з них $h^{sel}(x) = h_m^{weak}(x)$, де m обирається згідно з оптимізаційним критерієм. Фактично, ми використовуємо оцінку помилки еі кожного слабкого класифікатора $h_i^{weak} \in H^{weak}$, так що $m = argmin_i e_i$.

Сильний класифікатор : Заданий набір N слабких класифікаторів, сильний класифікатор обчислюється шляхом лінійної комбінації селекторів. Крім того, значення функції con f() (яка пов'язана з відступом) може бути інтерпретовано як показник впевненості сильного класифікатора.

$$hStrong(x) = sign(conf(x)) \tag{4.1}$$

$$conf(x) = \sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(x) \tag{4.2}$$

Основна ідея онлайн-підсилення полягає в введенні так званих селекторів. Вони ініціалізуються випадковим чином, і кожен з них має окремий набір слабких класифікаторів. Коли надходить новий навчальний приклад, слабкі класифікатори кожного селектора оновлюються. Найкращий слабкий класифікатор (з найнижчою помилкою) обирається селектором, де помилка слабкого класифікатора оцінюється на основі до цього часу побачених прикладів. Складність визначається кількістю селекторів.

Найбільшу частину часу обробки займає оновлення слабких класифікаторів. Щоб прискорити цей процес, ми пропонуємо модифікацію, яка полягає в використанні єдиного "глобального пула слабких класифікаторів" , який використовується всіма селекторами, замість окремих пулів для кожного з них. Перевагою цієї модифікації є те, що тепер для кожного надходження прикладу всі слабкі класифікатори потрібно оновити лише один раз. Потім селектори послідовно

переходять до найкращого слабкого класифікатора з урахуванням поточного оцінюваного значення λ, і вага важливості передається наступному селектору. Ця процедура повторюється до оновлення всіх селекторів. Нарешті, на кожному кроці часу доступний оновлений сильний класифікатор. Щоб збільшити різноманіття слабких класифікаторів і дозволити зміни в середовищі, найгірший слабкий класифікатор зі спільного пула функцій замінюється новим випадково вибраним.

Переваги: об'єкт відстежується досить точно, навіть якщо алгоритм вже застарів.

Недоліки: відносно низька швидкість роботи, висока чутливість до шуму і перешкод, та неможливість зупинити відстеження, коли об'єкт втрачено.

Рисунок 4.2 – Приклад роботи Boosting Tracker[20]

4.3.2 MIL Tracker

Одним з основних викликів, який часто не обговорюється, є вибір позитивних та негативних прикладів під час оновлення адаптивної моделі вигляду. Найчастіше це робиться шляхом використання поточного положення відстежувача як одного

позитивного прикладу та вибірки навколо положення відстежувача для негативних прикладів. Однак, якщо положення відстежувача не є точним, модель вигляду оновлюється з підоптимальним позитивним прикладом. З часом це може погіршити модель і спричинити відхилення. З іншого боку, якщо використовуються кілька позитивних прикладів (взятих з невеликого околу поточного положення відстежувача), модель може просто заплутатися.

Деякі з вищезазначених проблем виникають у задачі виявлення об'єктів, оскільки важко досягти консистентності у виборі позитивних прикладів для навчання. Іншими словами, точне розташування об'єктів невідоме. Фактично, автори MIL трекера стверджують, що виявлення об'єктів має вроджені неоднозначності, що робить навчання класифікатора складнішим за допомогою традиційних методів. З цієї причини вони пропонують використовувати підхід багаторазового навчання (Multiple Instance Learning, MIL) для виявлення об'єктів. Основна ідея цієї парадигми навчання полягає в тому, що під час навчання приклади представляються у вигляді наборів (часто називаних "мішками"), і мітки надаються для мішків, а не для окремих екземплярів. Якщо мішок має позитивну мітку, вважається, що він містить принаймні один позитивний екземпляр, в іншому випадку мішок є негативним. Наприклад, у контексті виявлення об'єктів позитивний мішок може містити кілька можливих прямокутників навколо кожного позначеного об'єкта (наприклад, лейбер, що позначає центр об'єкта, та алгоритм обрізає декілька прямокутників навколо цієї точки). Таким чином, неоднозначність передається до алгоритму навчання, який повинен визначити, який екземпляр у кожному позитивному мішку є найбільш "правильним". Хоча можна стверджувати, що ця задача навчання є складнішою в тому сенсі, що надається менше інформації для вчителя, з деяких позицій вона насправді є простішою, оскільки допускається деяка гнучкість у виборі рішення.

Переваги: більш стійкий до шуму, виявляється досить точно.

Недоліки: відносно низька швидкість і неможливість зупинки відстеження, коли об'єкт втрачається.

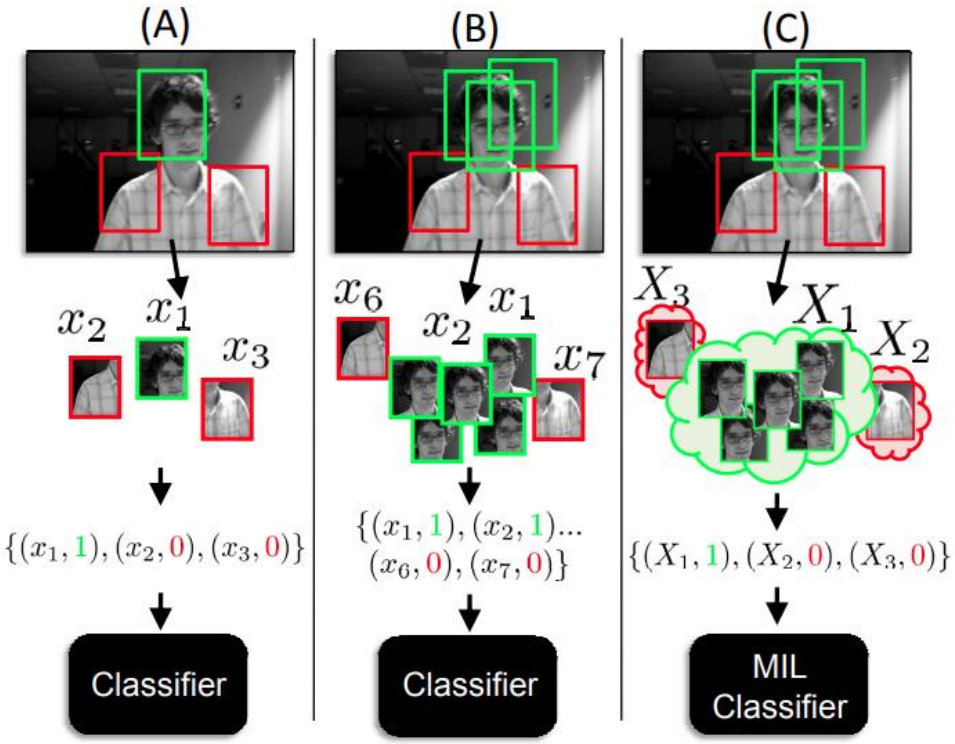

Рисунок 4.3 - Різниця між звичайними класифікаторами та MIL

Algorithm 1 MILTrack

Input: New video frame number k

1: Crop out a set of image patches, $X^s = \{x | s > ||l(x) - l^*_{t-1}||\}$ and compute feature vectors.
2: Use MIL classifier to estimate $p(y = 1|x)$ for $x \in X^s$.
3: Update tracker location $l^*_t = l\Big(\text{argmax}_{x \in X^s} p(y|x)\Big)$
4: Crop out two sets of image patches $X^r = \{x | r > ||l(x) - l^*_t||\}$ and $X^{r,\beta} = \{x | \beta > ||l(x) - l^*_t|| > r\}$.
5: Update MIL appearance model with one positive bag X^r and $|X^{r,\beta}|$ negative bags, each containing a single image patch from the set $X^{r,\beta}$

Рисунок 4.4 - Алгоритм роботи MIL трекеру

4.3.3 KCF (Kernelized Correlation Filters) Tracker

Це поєднання двох алгоритмів: BOOSTING і MIL. Концепція методу полягає в тому, що набір зображень з "мішка", отриманий за допомогою методу MIL, має багато областей, що перекриваються. Застосування кореляційного фільтрування до цих областей дозволяє відстежувати рух об'єкта з високою точністю та передбачити його подальше положення.

Фільтр навчається за допомогою перекладених (зсунутих) екземплярів патча цілі. Під час тестування оцінюється відповідь фільтра, і максимальне значення вказує на нове положення цілі. Фільтр навчається в режимі онлайн і оновлюється з кожним кадром, щоб трекер адаптувався до помірних змін цілі.

Основною перевагою трекера на основі кореляційного фільтра є обчислювальна ефективність. Це пояснюється тим, що обчислення можуть проводитись ефективно у просторі Фур'є. Таким чином, трекер працює у режимі суперреального часу з кількома сотнями кадрів в секунду.

Існують як лінійні, так і нелінійні (ядрові) версії трекера, які виводяться з єдиного принципу найменших квадратів.

Переваги: достатня швидкість та точність, зупинка відстеження при втраті відстежуваного об'єкта.

Недоліки: неможливість продовжити відстеження після втрати об'єкта.

4.3.4 TLD Tracker (Tracking Learning Detection)

Цей метод відображає високі показники і має успішне застосування у комерції. Він був розроблений для довготривалого відстеження різних класів цілей. У порівнянні з іншими алгоритмами, які фокусуються виключно на відстеженні об'єктів або лише на пошуку об'єктів у кадрі, цей метод поєднує обидва підходи і

поєднує їх на етапі навчання. TLD складається з трекера, який аналізує рух цілі від кадру до кадру, і детектора, який незалежно сканує кожен кадр послідовності.

При використанні детектора можуть виникати два види помилок: помилкові спрацьовування, коли об'єкти неправильно ідентифікуються, і помилкові відмови, коли об'єкти не розпізнаються правильно.

Навчальний компонент стежить за станом як трекера, так і детектора, і створює навчальний набір для зменшення кількості помилок у детекторі. При цьому навчальний компонент передбачає можливість одночасного неправильного функціонування як трекера, так і детектора.

Під час навчання детектор вдосконалюється у визначенні широкого спектру варіацій представлення цілей та у чіткому відокремленні цілей від тла. Особливий інтерес представляє запропонований навчальний компонент, відомий як P-N Learning. Його основна ідея полягає у використанні двох експертів: P-експерта, який виявляє тільки помилкові відмови, і N-експерта, який виявляє помилкові спрацьовування.

Навіть при наявності помилок експертів, їх незалежне використання компенсує ці недоліки. Детектор складається з бінарного класифікатора, який сканує вікно, і моделі цілі, що заснована на навчальних прикладах представлення цілі. Кожен елемент навчального набору обробляється окремо від інших. У разі наявності N вузлів у регулярній сітці сканування, для кожного кадру виникає 2N можливих комбінацій. Таким чином, може статися так, що одній цілі відповідає кілька можливих місць розташування, що порушує зв'язність руху. В таких випадках виникає ситуація "помилкового спрацьовування".

Основна оцінка помилок детектора ґрунтується на зв'язності руху, яка є важливим параметром. P-експерт використовує часову зв'язність і припускає, що ціль рухається вздовж траєкторії. Він використовує результати трекера для оцінки переміщення цілі і, якщо детектор відхиляє найбільш ймовірне місце розташування цілі (згідно з трекером), створює відповідний навчальний набір. Цей набір

генерується з поточної цілі шляхом зміщення, масштабування та повороту (близько 100 прикладів).

N-експерт використовує просторову зв'язність цілі і припускає, що ціль знаходиться лише в одному місці. Він також використовує дані трекера та знаходить найкраще зіпсоване місце серед результатів детектора, що оновлює стан трекера, а решту додає до навчального набору як негативні приклади. Нагадаємо, що модель цілі складається з колекції негативних та позитивних прикладів - областей зображення. Схожість між двома прикладами розраховується як нормалізована кореляція і виступає критерієм в класифікаторі найближчого сусіда (Nearest Neighbor).

Саме цей класифікатор приймає рішення щодо оновлення моделі. Детектор використовує скануюче вікно для пошуку відповідності зразку. При наявності великої кількості зразків, пошук може зайняти значну кількість часу.

Тому використовується каскад з трьох класифікаторів: класифікатор різниці моделей, класифікатор ансамблю та класифікатор найближчого сусіда. На практиці до останнього класифікатора надходить приблизно 50 екземплярів, що можна ефективно обробити за допомогою NN-класифікатора. У якості трекера використовується Median-Flow трекер, який є розширенням трекера Лукаса-Канаде і працює з оптичними потоками. Розширення включає в себе визначення втрати цілі: якщо різниця зміщення певної точки цілі та медіани всіх точок перевищує заданий поріг, то це вказує на втрату цілі. Це може статися при швидкому руху або перекритті цілі. Якщо ні детектор, ні трекер не надають розташування цілі на певному етапі, то ціль вважається втраченою. В іншому випадку вибирається найкращий результат, що базується на обчисленні кореляції з шаблоном. З таким підходом детектор використовує вже відомі шаблони, але може виявити ціль, якщо вона різко змінила положення або вийшла з тривалого перекриття, а трекер додає нові шаблони, оновлюючи базу детектора, і враховує просторово-часовий контекст.

Переваги: демонструє відносно хороші результати щодо стійкості до зміни розміру об'єкта та перекриття іншими об'єктами.

Недоліки: досить непередбачувана поведінка, нестабільність виявлення та відстеження, постійна втрата об'єкта, відстеження схожих об'єктів замість вибраного.

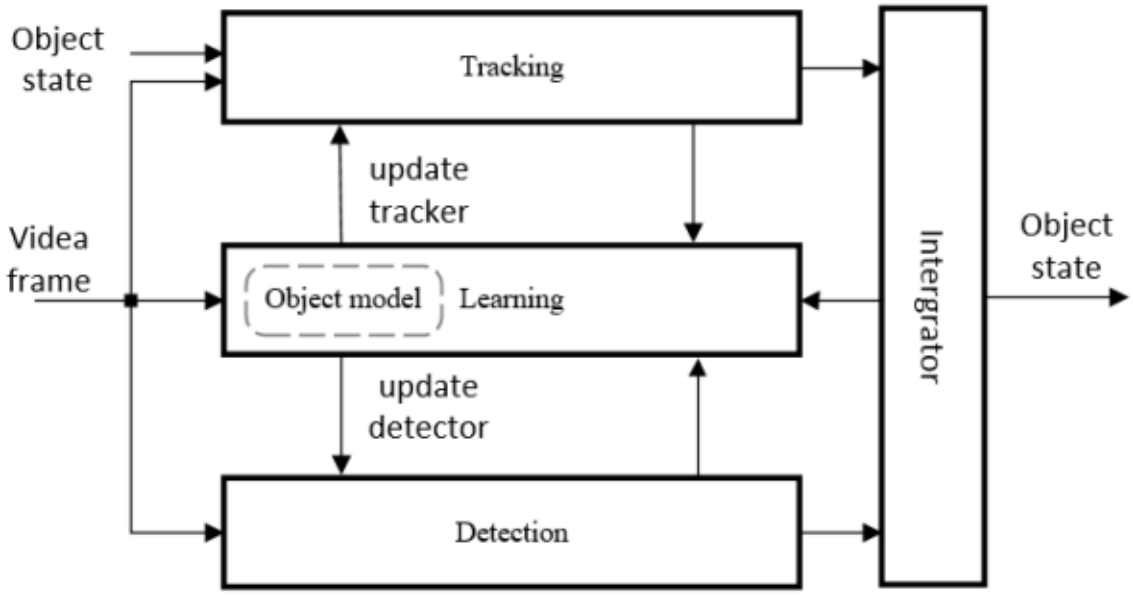

Рисунок 4.5 - Структура TLD алгоритму[21]

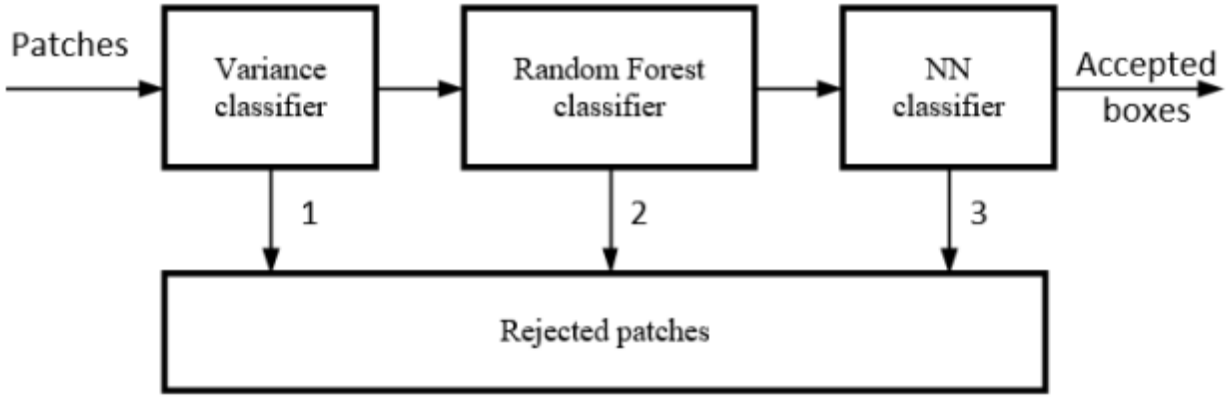

Рисунок 4.6 - Структура каскадного класифікатора[21]

4.3.5 MOSSE Tracker(Minimum Output Sum of Squared Error)

Цей алгоритм базується на обчисленні адаптивних кореляцій у просторі Фур'є. Фільтр мінімізує суму квадратних помилок між фактичним вихідним кореляційним сигналом та прогнозованим вихідним кореляційним сигналом. Цей відстежувач є стійким до змін освітлення, масштабу, положення та негнучких деформацій об'єкта.

Алгоритм MOSSE вніс технологію кореляційних фільтрів до області візуального відстеження. Цей тип алгоритму може пристосовуватись до проблем з перекриттям і обертанням та досягає вражаючої швидкості відстеження - 669 кадрів в секунду (fps). Запускаючись в 26 разів швидше, ніж передовий алгоритм MIL, фільтр MOSSE навчається за допомогою першого кадру і може демонструвати високу стійкість до змін освітлення, масштабу та положення об'єкта. Коли об'єкт перекритий, алгоритм може визначити статус відстеження об'єкта та оновити параметри фільтра відповідно до значення PSR. Коли об'єкт знову з'являється, його можна відстежувати знову.

У алгоритмі MOSSE для створення швидкого відстежувача використовується швидке перетворення Фур'є (FFT) для обчислення кореляції у Фур'є-просторі. Спочатку обчислюється двовимірне перетворення Фур'є вхідного зображення (F = F (f)) та фільтра (H = F (h)). Теорема про згортку стверджує, що кореляція є множенням елементів у Фур'є-просторі. Символ ⊗ позначає множення елементів поелементно, * позначає комплексне спряження, а представлення кореляції виглядає так:

$$g = f \otimes h \tag{4.3}$$

де g, f та h відповідають вихідному відгуку, вхідному зображенню та шаблону фільтра відповідно. Зауважимо, що нам потрібно лише визначити шаблон фільтра h, щоб отримати вихідний відгук. У рівнянні використовується швидке перетворення Фур'є (FFT). Тому операція згортки перетворюється на операцію

множення точок, що значно зменшує обсяг обчислень. Іншими словами, вищезазначена формула стає:

$$F(g) = F(f \otimes h) = F(f) \cdot F(h)^* \tag{4.4}$$

Потім вищезазначена формула ускорюється наступним чином: $G = F \cdot H^*$ і наступним завданням є відстеження та пошук шаблону фільтра H*.

$$H^* = \frac{G}{F} \tag{4.4}$$

У процесі фактичного відстеження ми повинні враховувати вплив факторів, таких як зовнішній вигляд об'єкта. Водночас, врахування m зображень об'єкта як посилання може значно покращити стійкість шаблону фільтра. Формула моделі MOSSE має такий вигляд:

$$min_{H^*} = \sum_{i=1}^{m} |H^* F_i - G_i|^2 \tag{4.5}$$

після серії перетворень, отримуємо рішення:

$$H^* = \frac{\sum_i G_i \cdot F_i^*}{\sum_i F_i \cdot F_i^*} \tag{4.6}$$

Алгоритм відстежує об'єкт використовуючи кореляцію фільтрів на вікні пошуку в наступному кадрі. Нове положення об'єкта представлено максимальним значенням відповідного вихідного сигналу. Потім виконується онлайн-оновлення в

новому положенні. Метод оновлення відстежувача використовує наступну формулу:

$$H_i^* = \frac{A_i}{B_i} \tag{4.7}$$

, де A та B – допоміжні матриці для обрахування частот фільтра

$$A_i = \eta G_i \otimes F_i^* + (1 - \eta) A_{i-1} \tag{4.8}$$

, де η – швидкість навчання під час трекінгу

$$B_i = \eta F_i \otimes F_i^* + B_{i-1} \tag{4.9}$$

І нарешті поріг відклику :

$$PSR = \frac{peak - \mu}{\sigma} \tag{4.10}$$

У експерименті вважається, що значення PSR від 20 до 60 є показником гарного ефекту відстеження. Якщо значення PSR менше 7, вважається, що відстеження не вдалося, і шаблон не оновлюється.

В цілому, алгоритм MOSSE може адаптуватися до незначних змін масштабу, але не в змозі адаптуватися до значних змін масштабу.

Переваги: дуже висока швидкість відстеження, більш успішне продовження відстеження об'єкта, якщо він був втрачений.

Недоліки: висока ймовірність продовження відстеження, якщо об'єкт втрачено і не з'являється в кадрі.

4.3.6 MedainFlow Tracker

Відстежувач отримує пару зображень It, It+1 та обмежувальну рамку βt і повертає обмежувальну рамку βt+1. Набір точок ініціалізується на прямокутній сітці в межах обмежувальної рамки βt. Потім ці точки відстежуються іншим трекером, який генерує потік руху між It та It+1.

Якість прогнозів точок оцінюється і кожній точці призначається помилка FB(Forward-BackwardError). 50% найгірших прогнозів відфільтровуються. Залишені прогнози використовуються для оцінки зсуву всієї обмежувальної рамки.

Для оцінки масштабу обчислюється співвідношення відстані між точками у поточному кадрі та попередньому кадрі. Медіана зміни масштабу по набору точок і передбачає зміну масштабної складової у кожному кадрі.

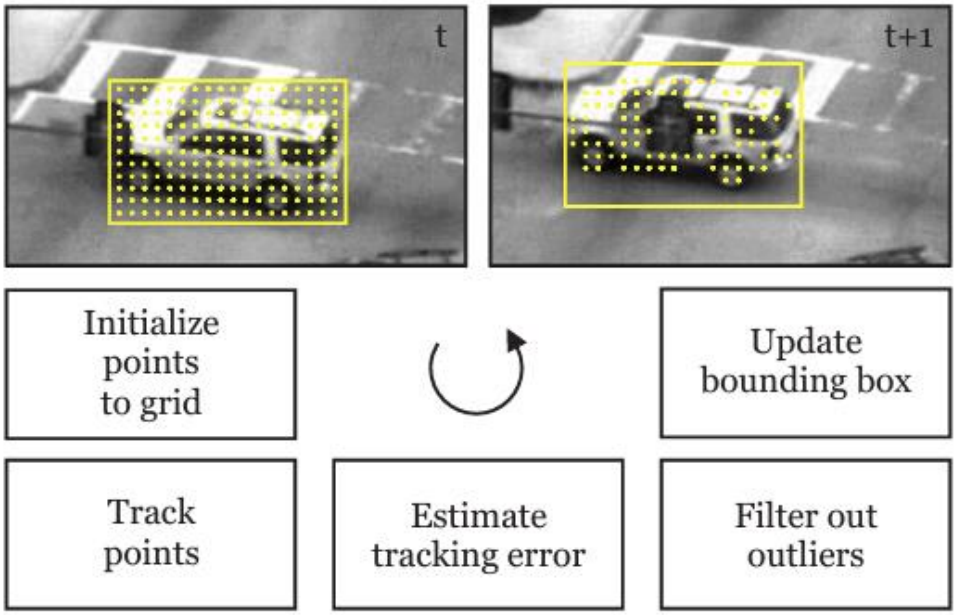

Рисунок 4.7 - Алгоритм роботи MedianFlow Tracker[19]

Детальніше про Forward-Backward error (FB). Нехай S = (I_1, I_2, ..., I_k) буде послідовністю зображень, а x_t буде місцезнаходженням точки в момент часу t. За допомогою довільного відстежувача точка x_t відстежується вперед на k кроків. Отримана траєкторія буде Tk_{tk} = (x_t, x_{t+1}, ..., x_{t+k}), де f вказує на вперед, а k позначає довжину. Нашою метою є оцінити помилку (надійність) траєкторії Tk_{tk}, враховуючи послідовність зображень S. З цією метою спочатку будується перевірочна траєкторія. Точка x_{t+k} відстежується назад до першого кадру і формує траєкторію Tk_b = ($\hat{x}_t$, $\hat{x}_{t+1}$, ..., $\hat{x}_{t+k}$), де $\hat{x}_{t+k} = x_{t+k}$. Forward-Backward error визначається як відстань між цими двома траєкторіями: FB(Tk_{tk}|S) = distance(Tk_{tk}, Tk_b).

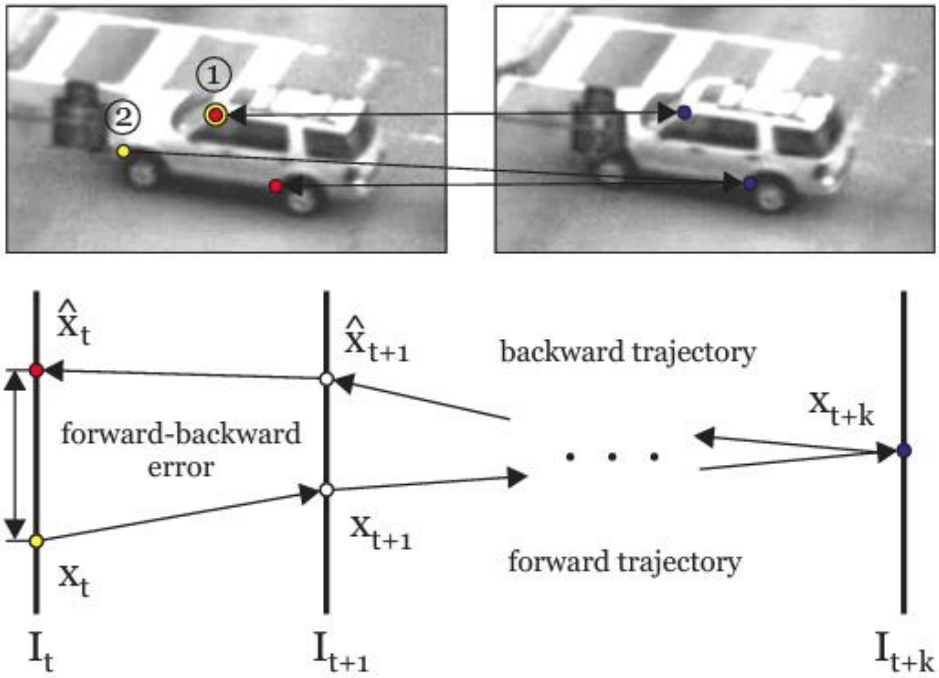

Рисунок 4.8 - Приклад Forward-Backward error та побудови траєкторій[19]

Переваги: досить висока швидкість та точність відстеження, якщо об'єкт не перекривається іншими об'єктами і його швидкість руху не є надто високою. Алгоритм досить точно визначає втрату об'єкта.

Недоліки: висока ймовірність втрати об'єкта при високій швидкості його руху.

5 РЕЗУЛЬТАТИ ТЕСТУВАННЯ АЛГОРИТМІВ ЗАХОПЛЕННЯ ЦІЛІ РАЗОМ ІЗ ПОПЕРЕДНЬОЮ ФІЛЬТРАЦІЄЮ ТА БЕЗ НЕЇ

5.1 Результати роботи алгоритмів трекінгу без фільтрації.

Для роботи було обрано для тестування наступні алгоритми трекінгу цілей :

1. MOSSE
2. Boosting
3. MIL
4. KCF
5. TLD
6. MedianFlow

Алгоритми були реалізовані та оцінювались на наступними критеріями:

1. FPS. Плавність кадру отриманого під часу процесу спостереження.
2. Відсоток захоплення цілі. Наскільки правильно трекер обирав ціль та чи не виходила вона за межі цільової області.
3. Avarage Peak Memory. Як багато оперативної пам’яті зайняв алгоритм.
4. Неперервне супроводження. Чи трекався об'єкт упродовж усього тестового відео.

Тестування проводилось на відеоматеріалі дрона-Шахіда, що був ліквідований 04.05.2023 над центром столиці. Програмний код наведений у Додатку А

Результати занесені у таблицю 5.1

Таблиця 5.1 – Результати тестування трекерів

Критерій / Трекер	FPS	% захоплення цілі	average peak memory [MB]	Неперервне супроводження
MOSSE	156	80	2,04	Так
Boosting	78	85	2,15	Так
MIL	50	82	1,01	Ні
KCF	260	80	1,78	Ні
TLD	50	60	1,027	Ні
MedianFlow	499	80	4,132	Так

Важливим фактором при обранні тестування алгоритма разом із попередньою фільтрацією буде саме використання оперативної пам’яті. Оскільки бойові місії FPV дрона не дозволяють йому встановити потужні мікросистеми, тому найкращим рішенням буде Алгоритм , що забезпечує мінімальне використання пам’яті та гарні показники , щодо відсотку захоплення цілі та FPS.

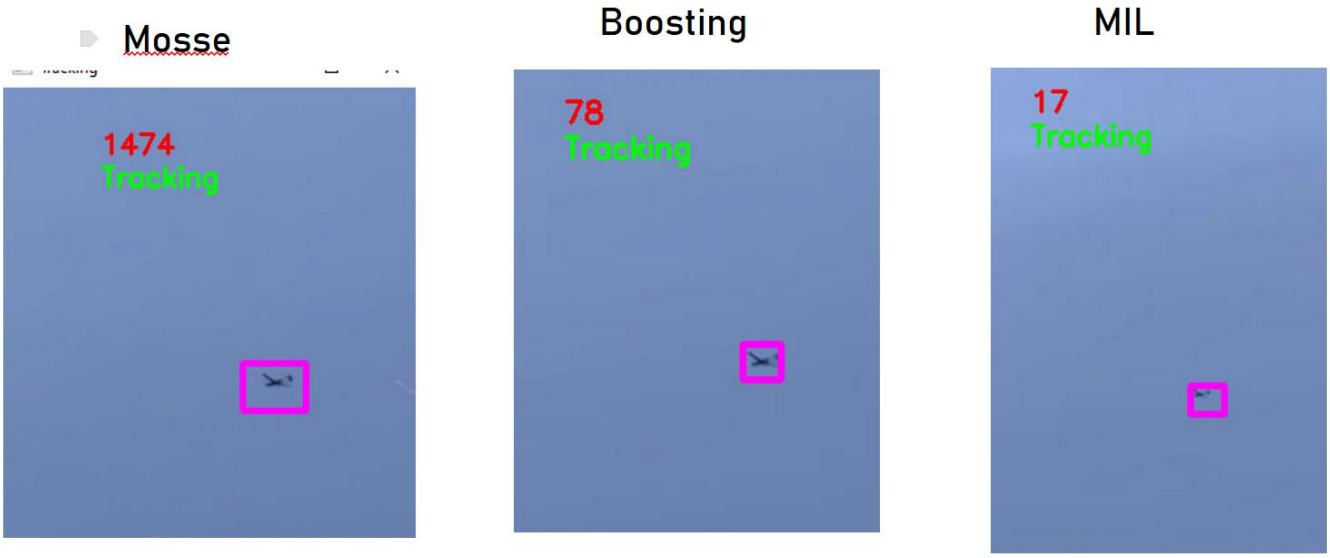

Рисунок 5.1 – Результати роботи трекерів

Вище наведені скриншоти – результати роботи різних трекерів. Бачимо що декілька з них не впоралися зі своєю задачею тому мають бути оптимізовані.

5.2 Оптимізація алгоритму захоплення цілі шляхом попередньої фільтрації

З огляду на невелику обчислювальну ефективність та високий відсоток захоплення цілі обираємо MIL Tracker, наступні експерименти будуть проведені задля покращення результатів цього трекеру шляхом вдалої попередньої фільтрації зображення.

На кожному етапі виявлення будемо робити попередню фільтрацію зображення. Це допомогло MIL трекеру краще обробляти зображення оскільки деякі типи фільтрації дали чіткіші контури зображення або ж прибрали непотрібні світлові відблисики тощо.

Результати захоплення цілі MIL трекером разом із різними попередніми фільтраціями наведені у таблиці 5.2.

Таблиця 5.2 – Тестування MIL Tracker разом з попередньою фільтрацією

Критерій / Фільтр	FPS	% захоплення цілі	average peak memory [MB]	Неперервне супроводження
Gauss Blur	15	55	1,01	Так
Median	16	50	1,03	Так
Gamma Corr	15	64	1,022	Ні
Laplasian	17	95	1,034	Так
Log transf.	11	50	1,57	Ні
Thresholding	18	100	0,68	Так

Бачимо , що деякі методи фільтрації дали значні покращення нашому алгоритму трекінгу. Виділятимуться з цього списку 2 методи – Збільшення різкості Лапласіаном(Laplasian) та метод Порогової обробки(Thresholding). Це відбулося за рахунок того, що саме ці два методи дозволили чіткіше алгоритму бачити контури зображення отож і трекінг відбувся результативніше. На 50 спробах MILTracker ні разу не згубив ціль з спостереження та захоплював близько 100% потрібного об’єкта. Чого не було без попередньої фільтрації.

Натомість Методи Гауса, Медіанний, Гамма кореляції та Логарифмічної трансформації більше працювали з кольором та освітленням зображення, що не дало значних покращень трекеру, оскільки конури потрібного об’єкта були не чіткими.

Результати роботи трекеру з різними попередніми фільтраціями скриншотами наведені на рисунку 5.2 та 5.3.

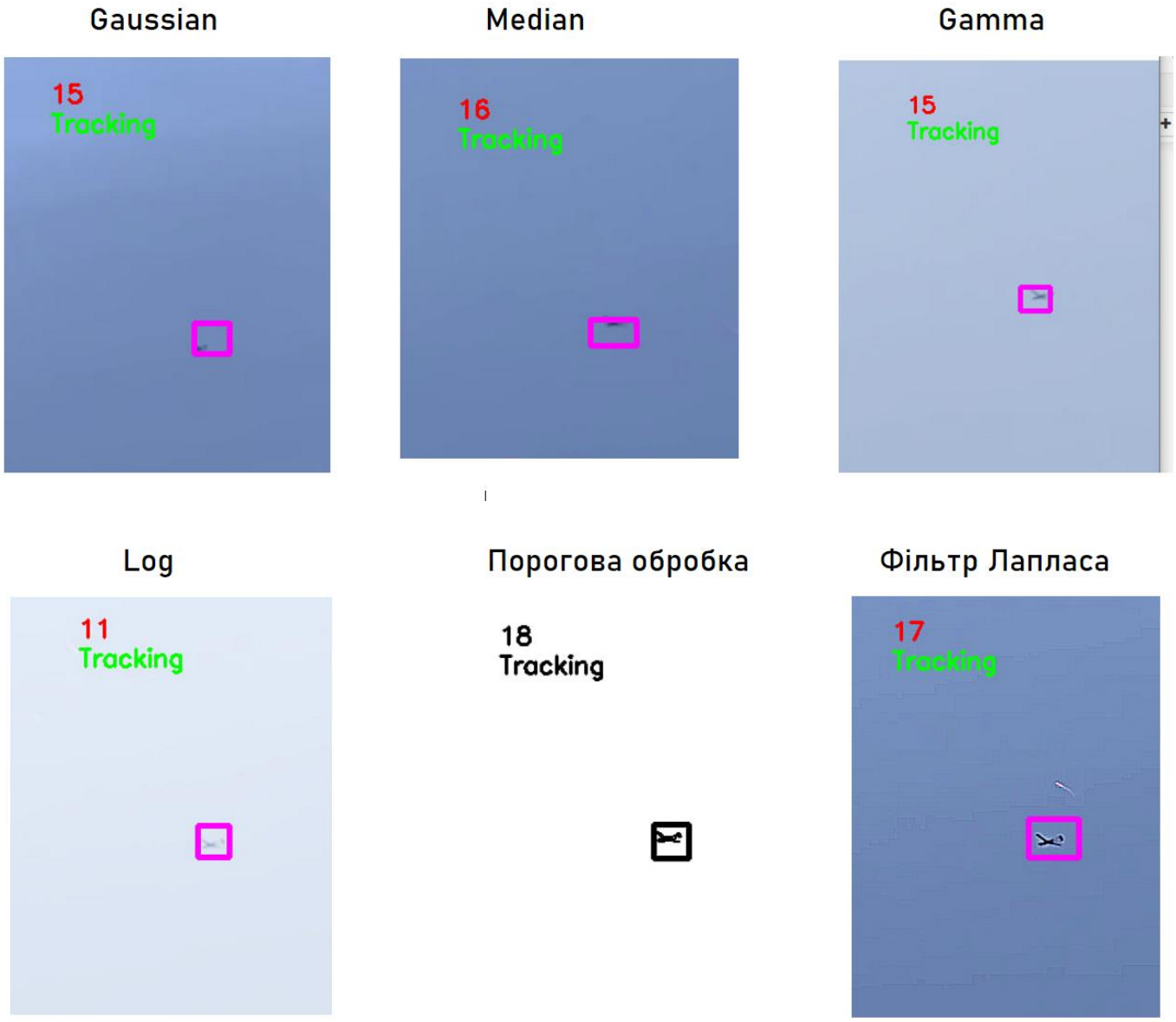

Рисунок 5.2 - Результати роботи трекерів разом з попередньою фільтрацією Об’єкт спостереження – дрон - Шахід.

Рисунок 5.3 - Результати роботи трекерів разом з попередньою фільтрацією. Об’єкт спостереження – танк.

6 ЗАГАЛЬНІ ВИСНОВКИ

У роботі було опрацьовано проблеми сучасних систем спостереження, що дало можливість коректно поставити та реалізувати задачу ефективного алгоритму захоплення цілі без безпосередньої участі оператора у реальному часі. В його основі лежать метод трекінгу MIL Tracker та спосіб попередньої фільтрації за допомогою Лапласіана.

Практична частина робота дала змогу оцінити вплив кожної фільтрації на відстежувач та визначити комбінацію фільтра та трекера з найкращими показниками відсотка цілі, що відстежується, та неперервного супроводження. Тестування проводилося на відеопотоках з реальними бойовими сценаріями за участі танків та дронів.

За допомогою попередньої фільтрації зображення вдалося домогтися кращої результативності від вже існуючих трекерів. Робота містить порівняння різних відстежувачів між собою без фільтрації та разом з нею.

Результати цього дослідження можуть бути використані для сучасних систем спостереження FPV дронами. Зокрема для знешкодження танків, проведення розвідувальних місій, підтримки вогню, рятувальних операцій, збору інформації про терористичні загрози та ураження вразливих місць бойових машин.

7 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 В. Г. Дзюба, А. Ю. Варфоломєєв. Курс лекцій з дисципліни «Системи технічного зору» для студентів спеціальностей 7.091001 та 8.091001 – «радіоелектронні апарати та засоби» : НТУУ «КПІ», 2011 р. – 153 с.

2 R. Haralick and L. Shapiro *Computer and Robot Vision*, Vol. 1, Addison-Wesley Publishing Company, 1992, pp 346 - 351.

3 S. K. Pal, A. Pramanik, J. Maiti, and P. Mitra, “Deep learning in multiobject detection and tracking: state of the art,” Applied Intelligence, vol. 51, no. 9, pp. 6400–6429, 2021.

4 A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in 2016 IEEE international conference on image processing (ICIP), pp. 3464–3468, IEEE, 2016.

5 A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, “Online multi-target tracking using recurrent neural networks,” in Thirty-First AAAI conference on artificial intelligence, 2017.

6 Fisher, Perkins, Walker & Wolfart *(2003)*. "Spatial Filters - Laplacian of Gaussian". *Retrieved 2010-09-13*.

7 Erik Reinhard. *High dynamic range imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann, 2006, pp. 233–234.

8 Shapiro, L. G. & Stockman, G. C: "Computer Vision", page 137, 150. Prentice Hall, 2001

9 "FPV Goggles For Drones". *DroneZon.com. Retrieved May 9, 2015*

10 Getreuer, Pascal (17 December 2013). "ASurvey of Gaussian Convolution Algorithms"

11 Ning J., Zhang L., Zhang D., and Wu C., “Robust Mean-Shift Tracking with Corrected Background-Weighted Histogram,” IET Computer Vision, vol. 6, no. 1, pp. 62-69, 2012.

12 Ross D., Lim J., Lin R., and Yang M., “Incremental Learning for Robust Visual Tracking,” International Journal of Computer Vision, vol. 77, no. 1, pp. 125-141, 2008

13 R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. PAMI, 27(10):1631– 1643, 2005.

14 Z. Kalal, J. Matas, and K. Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. CVPR, 2010.

15 H. Wu, A. C. Sankaranarayanan, and R. Chellappa. In situ evaluation of tracking algorithms using time reversed chains. CVPR, 2007.

16 B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. IJCAI, 81:674–679, 1981

17 https://www.researchgate.net/figure/Gamma-correction-curve-for-different-values-of-gdocumentclass12ptminimal_fig2_329571120

18 https://arijitkar98.github.io/2018/07/09/second-coding-phase.html

19 Zdenek Kalal, K. Mikolajczyk, Jiri Matas" Forward-Backward Error: Automatic Detection of Tracking Failures" Published 23 August 2010 Computer Science, Engineering 2010 20th International Conference on Pattern Recognition

20 https://broutonlab.com/blog/opencv-object-tracking

21 Zdenek Kalal, Krystian Mikolajczyk, Jiri Matas “Tracking-Learning-Detection” ieee transactions on pattern analysis and machine intelligence,vol.6,no.1,january2010

ДОДАТОК А

Лістинг програми, що реалізує алгоритм захоплення цілі разом із попередньою фільтрацією.

```
import cv2
import tracemalloc
import numpy as np

cap = cv2.VideoCapture("drone2.MOV")

tracker = cv2.legacy.TrackerMIL_create()
success, img = cap.read()
kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
img = cv2.filter2D(img, -1, kernel)

# select a bounding box ( ROI )
bbox = cv2.selectROI("Tracking", img, False)
tracker.init(img, bbox)


def drawBox(img, bbox):
    x, y, w, h = int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 255), 3, 1)
    cv2.putText(img, "Tracking", (75, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
255, 0), 2)

tracemalloc.start()
current_memories = []
peak_memories = []
while True:
    timer = cv2.getTickCount()
    success, img = cap.read()
    img = cv2.filter2D(img, -1, kernel)

    success, bbox = tracker.update(img)

    if success:
        drawBox(img, bbox)
    else:
        cv2.putText(img, "Loss", (75, 75), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
0, 255), 2)

    fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer)
    cv2.putText(img, str(int(fps)), (75, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(0, 0, 255), 2)
    cv2.imshow("Tracking", img)

    key = cv2.waitKey(30)
    if key == 27:
        break
```

```
current, peak = tracemalloc.get_traced_memory()
current_memories.append(current/(1024*1024))
peak_memories.append(peak/(1024*1024))
tracemalloc.reset_peak()
# tracemalloc.clear_traces()
del current, peak
print('Average current memory [MB]: {}, average peak memory [MB]: {} +/-
{}'.format(
      round(np.mean(current_memories), 4), round(np.mean(peak_memories), 4),
      round(np.std(peak_memories), 4))
)

# stopping the library
tracemalloc.stop()

cv2.destroyAllWindows()
```