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

Радіотехнічний факультет

Кафедра радіотехнічних систем

«На правах рукопису»
УДК 621.396

До захисту допущено:

Завідувач кафедри

_____ Сергій ЖУК

«_9_»____10_____2024р.

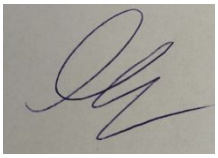
Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Радіоелектронна інженерія» зі спеціальності 172 «Телекомунікації та радіотехніка»

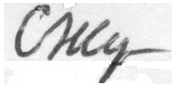
на тему: **«Алгоритми траєкторної обробки інформації за даними мережі відеодатчиків»**

Виконав:
студент ІІ курсу, групи РС-21мн
Кот Максим Геннадійович



_____

Науковий керівник:
д.т.н., завкафедри РТС
Жук Сергій Якович



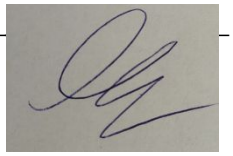
_____

Рецензент:

д.т.н., професор кафедри РІ

Шарпан Олег Борисович _____

Засвідчую, що у цій магістерській дисертації немає запозичень
з праць інших авторів без відповідних посилань.
Студент _____



Київ – 2024 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Радіотехнічний факультет

Кафедра радіотехнічних систем

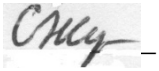
Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-наукова програма «Радіоелектронна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

 _ Сергій ЖУК

«_9__»____10_____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Коту Максиму Геннадійовичу

1. Тема дисертації «Алгоритми траєкторної обробки інформації за даними мережі відеодатчиків», науковий керівник дисертації Жук Сергій Якович, професор, д.т.н., затверджені наказом по університету від «28»____03__2024 р. №1483-с

2. Термін подання студентом дисертації «_7_»__06_____ 2024 р. _____

3. Об'єкт дослідження _процеси виявлення і оцінювання параметрів руху БПЛА за даними__мережі__відеодатчиків_____

_____

4. Предмет дослідження _методи виявлення і оцінювання параметрів руху БПЛА за_даними__мережі__відеодатчків_____

_____

5. Перелік завдань, які потрібно розробити _ алгоритм виявлення рухомого об'єкту на відео та провести його аналіз, розробити алгоритм визначення координат рухомого об'єкту за даними двох відеокамер, розробити алгоритм фільтрації параметрів руху об'єкту за даними двох відеокамер, виконати аналіз ефективності розробленого алгоритму траєкторної фільтрації та можливостей його практичної реалізації, розробити стартап проект._____

_____

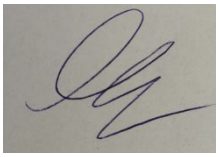
6. Орієнтовний перелік графічного (ілюстративного) матеріалу презентація_у_вигляді_слайдів_____

7. Орієнтовний перелік публікацій тезиси доповідей на науково-технічних конференціях_____

9. Дата видачі завдання 17. 10. 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз алгоритмів виявленні рухомих цілей	17.11.2023	
2	Розробка алгоритму виявлення цілей	20.12.2023	
3	Розробка алгоритму визначення координат по даним мережі датчиків	10.01.2024	
4	Розробка алгоритму траєкторної фільтрації	10.03.2024	
5	Аналіз можливостей практичної реалізації	10.04.2024	
6	Стартап проект	10.05.2024	

Студент  Максим КОТ

Науковий керівник Сергій ЖУК

РЕФЕРАТ

Магістерська дисертація «Алгоритми траєкторної обробки інформації за даними мережі відеодатчиків» займає 74 сторінок, 89 формул, 42 рисунки та 6 таблиць.

Дана дисертація присвячена розробці алгоритмів виявлення і оцінювання параметрів руху БПЛА за даними мережі відеодатчиків.

Мета дослідження: розробити алгоритми виявлення і оцінювання параметрів руху БПЛА за даними мережі відеодатчиків.

Об'єкт дослідження: процеси виявлення і оцінювання параметрів руху об'єктів за даними мережі відеодатчиків.

Предмет дослідження: методи виявлення і оцінювання параметрів руху об'єктів за даними мережі відеодатчиків.

Обґрунтовано актуальність задач виявлення і оцінювання параметрів руху БПЛА за допомогою мережі відеодатчиків. Проведено порівняння алгоритмів виявлення об'єктів, які рухаються, на зображеннях: метод оптичного потоку, Camshift, Meanshift та метод міжкадрової різниці. Розроблено алгоритм виявлення рухомого об'єкту на відео з використанням міжкадрових різниць і проведено його аналіз на модельному і реальному відеозображеннях.

З використанням тріангуляційного методу розроблено алгоритм визначення координат рухомого об'єкту за даними двох відеокамер у прямокутній системі координат та проведено аналіз його точносних характеристик. На основі методу калманівської фільтрації розроблено алгоритм оцінювання параметрів руху об'єкту за даними двох відеокамер у прямокутній системі координат та виконано аналіз його ефективності шляхом статистичного моделювання. Розглянуто можливості практичної реалізації розробленого алгоритму траєкторної обробки інформації на сучасній цифровій базі.

Ключові слова: БПЛА, відеокамера, зображення, міжкадрова різниця, фільтр Калмана, триангуляція, параметри руху, оцінювання, точносні характеристики.

ABSTRACT

The master's thesis "Algorithms for the trajectory processing of information based on the data of the network of video sensors" occupies 74 pages, 89 formulas, 42 figures and 6 tables.

This dissertation is devoted to the development of algorithms for detecting and evaluating UAV movement parameters based on video sensor network data.

The purpose of the study: to develop algorithms for detecting and evaluating UAV movement parameters based on video sensor network data.

The object of the study: the processes of detection and evaluation of the parameters of the movement of objects based on the data of the network of video sensors.

The subject of the research: methods of detecting and evaluating the parameters of the movement of objects based on the data of a network of video sensors.

The relevance of the tasks of detecting and evaluating UAV movement parameters using network of video sensors is substantiated. Algorithms for detecting moving objects in images are compared: the optical flow method, Camshift, Meanshift, and the interframe difference method. Algorithm for detecting moving object on video using inter-frame differences was developed and its analysis was carried out on model and real video images.

Using triangulation method, algorithm for determining coordinates of moving object based on the data of two video cameras in a rectangular coordinate system was developed, and its accuracy characteristics were analyzed. Based on the Kalman filtering method, algorithm was developed for estimating object's movement parameters based on the data of two video cameras in rectangular coordinate system, and its effectiveness was analyzed by statistical modeling. Possibilities of practical implementation of the developed algorithm of trajectory processing of information on a modern digital base are considered.

Keywords: UAV, video camera, image, interframe difference, Kalman filter, triangulation, motion parameters, evaluation, accuracy characteristics.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БПЛА – безпілотний літальний апарат

ЕОМ – електронно-обчислювальна машина

СКВ – середнє квадратичне відхилення

СК – система координат

ВСТУП

Останнє десятиріччя продемонструвало бурхливий розвиток БПЛА [1]. Вони отримали широке застосування в багатьох сферах цивільних операцій, дистанційного зондування, моніторингу дорожнього руху в режимі реального часу, досліджень і порятунку, забезпечення бездротового покриття, розподілу матеріалів, безпеки та спостереження, в сільському господарстві та перевірки цивільної інфраструктури.

Комерційна індустрія БПЛА, які також широко відомі як дрони, за останні роки значно зросла, завдяки чому ці пристрої стали доступними для громадськості. Продажі дронів постійно зростають з кожним роком, і очікується, що в майбутньому вони будуть набагато більш поширеними.

В той же час експоненційно зростаюча публічна доступність дронів створює велику загрозу загальній безпеці та конфіденційності. В якості прикладів можна навести наступні події: інцидент безпеки навколо Білого дому, таємнича поява безпілотників протягом кількох днів навколо атомних електростанцій у Франції, жахливе майже зіткнення авіалайнера та дрона поблизу міжнародного аеропорту Лос-Анджелеса, а також вторгнення дрона опозиційної партії під час кампанії канцлера Німеччини, що попередило співробітників служби безпеки. Це призводить до необхідності розробки систем безпеки, які вирішують завдання виявлення, визначення місця розташування і параметрів руху БПЛА. На сьогодні в провідних країнах світу вирішення цієї задачі виведено на рівень національної безпеки.

Одним з основних засіб виявлення і супроводження малих БПЛА є системи відеоспостереження. У порівнянні з радіолокаційними засобами вони забезпечують також ефективне розпізнавання типів БПЛА. На сьогодні, з поширенням цифрових каналів спостереження на основі ПЗЗ-матриць, розвитком обчислювальної техніки, а також бездротових каналів зв’язку з'явилася можливість створювати мережі відеоспостереження для локації об'єктів у реальному режимі часу. Такі мережі є різновидом бездротових сенсерних мереж, в яких реалізується кутомірний метод.

Мета дослідження: розробити алгоритми виявлення і оцінювання параметрів руху БПЛА за даними мережі відеодатчиків.

Об’єкт дослідження: процеси виявлення і оцінювання параметрів руху об’єктів за даними мережі відеодатчиків.

Предмет дослідження: методи виявлення і оцінювання параметрів руху об’єктів за даними мережі відео датчиків.

Для досягнення мети в роботі вирішено наступні часткові завдання:

1. Розробка алгоритму виявлення рухомого об’єкту на відео з використанням міжкадрових різниць та його аналіз.
2. Розробка алгоритму визначення координат рухомого об’єкту за даними двох відеокамер у прямокутній системі координат.
3. Розробка алгоритму фільтрації параметрів руху об’єкту за даними двох відеокамер у прямокутній системі координат.
4. Аналіз ефективності розробленого алгоритму траєкторної фільтрації шляхом статистичного моделювання
5. Аналіз можливості практичної реалізації розробленого алгоритму траєкторної обробки інформації

1 АКТУАЛЬНІСТЬ ЗАДАЧ ВИЯВЛЕННЯ І ОЦІНЮВАННЯ ПАРАМЕТРІВ РУХУ БПЛА ЗА ДОПОМОГОЮ МЕРЕЖІ ВІДЕОДАТЧИКІВ

БПЛА, також відомі як дрони, стрімко завойовують світ, стаючи незамінними помічниками в найрізноманітніших сферах діяльності. Від поля бою до рятувальних операцій, від логістичних маршрутів до фермерських угідь — БПЛА беруть на себе все нові й нові завдання, демонструючи свої можливості та ефективність. Завдяки їм можна здійснювати розвідку, доставку вантажів, моніторинг сільськогосподарських угідь, а також надавати допомогу в надзвичайних ситуаціях. Їхня універсальність і багатофункціональність роблять їх незамінними інструментами в сучасному світі.

У військовій сфері дрони використовуються для розвідувальних операцій, спостереження за ворогом, а також для точкових ударів. Вони можуть літати на великій висоті, залишаючись непомітними, і надавати військовим точну інформацію в режимі реального часу. Це значно підвищує ефективність військових операцій і знижує ризики для людських життів. У рятувальних операціях дрони використовуються для пошуку зниклих людей, доставки медичних засобів у важкодоступні райони та оцінки масштабів катастроф. Вони можуть швидко облітати великі території і передавати зображення рятувальним командам, що дозволяє швидше і точніше координувати дії.

У сільському господарстві БПЛА використовуються для моніторингу стану посівів, внесення добрив і пестицидів, а також для картографування полів. Вони можуть літати над полями і збирати дані про стан рослин, що дозволяє фермерам оптимізувати використання ресурсів і підвищувати врожайність.

У логістиці дрони використовуються для доставки товарів і вантажів, особливо в умовах, де традиційні методи доставки недоступні або неефективні. Вони можуть швидко і безпечно доставляти пакунки у важкодоступні райони, знижуючи час і вартість доставки.

Однак, ця всебічна інтеграція БПЛА у наше життя несе з собою і численні виклики. Зростання їхньої популярності робить вкрай актуальним питання ефективного виявлення та відстеження цих літальних апаратів. Різке збільшення

кількості дронів в повітряному просторі ускладнює завдання контролю та забезпечення безпеки, як для цивільного, так і для військового сектору. Необхідно розробляти нові технології і методи, які дозволять ефективно ідентифікувати дрони, визначати їхні маршрути і запобігати потенційним загрозам.

Для таких сфер, як оборона стає критично важливим чітко й оперативно ідентифікувати БПЛА в режимі реального часу. Надійне виявлення та відстеження дронів дозволяє попереджати можливі загрози, координувати дії під час рятувальних операцій, а також ефективніше використовувати ресурси. У сфері оборони це дозволяє забезпечити захист від ворожих дронів, які можуть використовуватися для розвідки або атак. У сфері логістики це дозволяє краще координувати доставку вантажів і знижувати ризик втрат. У рятувальних операціях це дозволяє швидше знаходити зниклих людей і доставляти допомогу.

Саме тому розробка високопродуктивних та надійних методів виявлення і відслідковування рухомих дронів перетворюється на пріоритетний напрямок досліджень, який має ключове значення для безпеки та ефективного використання технологій БПЛА.

Нові алгоритми, сенсори та системи штучного інтелекту дозволяють значно підвищити точність і швидкість виявлення дронів, знижуючи ризик помилкових тривог і забезпечуючи своєчасне реагування на загрози.

Інтеграція цих технологій у різні системи і платформи забезпечує комплексний підхід до вирішення проблеми і дозволяє ефективно використовувати можливості дронів у найрізноманітніших умовах.

Класифікація БПЛА за різними критеріями дає змогу краще зрозуміти їхні можливості та сфери застосування:

Цивільна сфера

- Фотографія
- Будівництво
- Доставка
- Логістика
- Спостереження
- Управління стихійними лихами

Сфери, які пов’язані з навколишнім довкіллям

- Моніторинг якості повітря
- Моніторинг ґрунтів
- Моніторинг посівів
- Гірська інспекція

Оборонна сфера

- Забезпечення медичними засобами в зоні бойових дій
- Шпигунство
- Спостереження за кордоном
- Скидання бомб

Класифікація за типом двигуна:

- БПЛА з електродвигуном: найбільш поширений тип, завдяки простоті конструкції, низькому рівню шуму та економному споживанню енергії.
- БПЛА з бензиновим двигуном: потужніші та мають більший запас ходу, але й шумніші та викидають шкідливі речовини.
- Гібридні БПЛА: поєднують переваги електродвигуна та бензинового, забезпечуючи баланс між потужністю, дальністю польоту та екологічністю.

Класифікація за будовою:

Одноротoрний (або гелікоптер): ця категорія БПЛА злітає та приземляється вертикально. Як правило, вони використовують несучий гвинт для контролю положення (крену, тангажу та повороту), а хвостовий ротор – для керування напрямком. Його головною перевагою є здатність перевозити важкі корисні вантажі протягом тривалого часу польоту. Однак складність їх механічних систем, а також великі розміри і дорожнеча роторів становлять небезпеку для безеквіпажних версій. Він більш нестійкий в погану погоду.

Багатороторний (або мультикоптер): це БПЛА, що має більше двох роторів. Ця категорія БПЛА далі підрозділяється на п’ять підкатегорій: біротор, триротор, квадротор, гексаротор і восьмиротор (октокоптер). Як і одноротoрні БПЛА,

багатороторні БПЛА також забезпечують вертикальний зліт і посадку. Вони швидкі та спритні в польоті, що дозволяє їм виконувати складні маневри та польоти в обмеженому просторі. Однак основним недоліком цих типів літаків є короткий час польоту.

З нерухомим крилом: принцип навігації цієї категорії БПЛА базується на простій конструкції нерухомого жорсткого крила. Класифікація цих дронів базується не тільки на типі крила, а й на корпусі та системі живлення (Li-ion, Li-Po батареї або газові). Вони поділяються на чотири підкатегорії: нормальні, зігнуті назад, зігнуті вперед і дельта. Крім того, вони можуть перевозити більший корисний вантаж, ніж мультиротори. Недоліком цих БПЛА є обмежена маневреність у польоті, що не дозволяє виконувати складні маневри та літати над обмеженими просторами, а також необхідність наявності злітно-посадкової смуги для зльоту та посадки.

Гібрид: остання категорія все ще знаходиться на стадії розробки. Це вдосконалена версія, яка використовує як багатороторні, так і безпілотні літальні апарати з нерухомим крилом. Вони забезпечують хорошу маневреність і швидкість на далеких польотах. Вони можуть перевозити великі корисні навантаження і не потребують злітно-посадкової смуги. Основними недоліками є висока ціна, складна механіка, перехід з горизонтального до вертикального польоту, нижчі показники щодо стабільності польоту та обмеження діапазонів швидкостей.

Класифікація за розміром:

- **Міні-БПЛА:** компактні та легкі, призначені для розвідки на невеликих відстанях.
- **БПЛА середнього розміру:** універсальні, використовуються для різних завдань.
- **Великі БПЛА:** потужні та здатні нести значне корисне навантаження.

Класифікація за повною злітною масою:

Найбільш стандартизований опис даного класифікатора описано у стандарті НАТО 4670 у таблиці 1.1.

Таблиця 1.1 — Класифікація за злітною масою за стандартом НАТО 4670[2]

Клас	Категорія	Рівень воєнни дій	Висота застосуван ня	Радіус дії	Рівень застосуван ня
Класс III (> 600 кг)	Ударні	стратегічн ий	до 20 000 м	Театр воєнних дій	Reaper
	HALE	стратегічн ий	до 20 000 м	Театр воєнних дій	Global Hawk
	MALE	оперативн ий	до 14 000 м	Оперативн а група	Heron Bayraktar TB2
Класс II (150-600 кг)	тактичні	тактичний	до 5500 м	Бригада	Hermes 450
Класс I (< 150 кг)	малі (15 кг)	тактичне формуванн я	до 1500 м	Батальйон	Scan Eagle PD-2
	міні (15 кг)	тактичний підрозділ	до 900 м	рота, взвод, відділення	Skylark
	мікро (66 Дж)	тактичний підрозділ	до 60 м	взвод, відділення	Black Widow

Класифікація військових БПЛА в США

Група 1 (мікро-, міні тактичні) — від 0 до 9 кг, до 300 метрів над ґрунтом, основний представник — «RQ-11 Raven».

Група 2 (малі тактичні) — від 9.5 до 25 кг; до 1000 метрів над ґрунтом, представник — «Scan Eagle»

Група 3 (тактичні) — менш, ніж 600 кг, представник — «RQ-7 Shadow»

Група 4 (персистентні) — більш, ніж 600 кг; представник — «MQ-1B Predator»

Група 5 (пенетрувальні) — більш, ніж 600 кг; представник — «MQ-9 Reaper»

Класифікація за призначенням:

За призначенням:

- **Розвідувальні БПЛА:** збирають інформацію про місцевість, об'єкти та противника.
- **Ударні БПЛА:** атакують наземні та повітряні цілі.
- **Транспортні БПЛА:** доставляють вантажі в важкодоступні місця.
- **Цивільні БПЛА:** використовуються для фото- та відеозйомки, картографування, моніторингу довкілля тощо.

Існують різні методи виявлення рухомого БПЛА[3], а саме:

- акустичний
- тепловий
- радарний
- оптичний

Акустичне виявлення має серйозний недолік у неможливості аудіовиявлення у випадку сильного шума навколо детектора.

Виявлення за допомогою тепловізора також не ефективне через те, що більшість БПЛА виготовлено з пластику з електродвигунами, які випромінюють мало тепла, навіть менше ніж плати. Саме це робить цей метод неефективним.

До недоліків радарного способу виявлення відноситься неефективність роботи у місцевості із високими будівлями, деревами. Крім того, вони не виявляють малогабаритні БПЛА, які запускаються в безпосередній близькості або летять на малій висоті.

Оптичне виявлення БПЛА суттєво відрізняється від інших методів, таких як радарне чи радіовимірювання, особливо за останні роки. Це пояснюється зростанням популярності камер, що використовуються для виявлення, завдяки їхній малій вазі, простоті експлуатації та низькому енергоспоживанню. Крім того,

камери здатні збирати величезну кількість даних, що робить їх незамінними для забезпечення потрібної інформації для подальшого аналізу та вжиття заходів.

Одним із способів виявлення об'єктів, які перебувають у русі, в режимі реального часу є метод міжкадрових різниць. Цей метод має низку переваг, серед яких головною є його простота у впровадженні. Для його роботи не потрібні великі обчислювальні потужності ЕОМ, що дозволяє застосовувати його навіть на пристроях з обмеженими ресурсами.

Метод міжкадрових різниць дозволяє ефективно виявляти рухомі об'єкти без необхідності додаткових ресурсів, що робить його доступним і практичним у багатьох ситуаціях. Використання цього методу особливо актуальне в умовах, де швидкість та ефективність є ключовими вимогами, наприклад, у системах відеоспостереження, охорони або моніторингу руху.

Існують і інші підходи до виявлення об'єктів, такі як нейронні мережі, які також можуть застосовуватися для виявлення рухомих об’єктів. Проте, вони потребують значно більше часу і обчислювальних ресурсів для своєї роботи, що може бути недоцільним у деяких випадках.

Нейронні мережі вимагають значного тренування і налаштування, а також високої продуктивності апаратного забезпечення, що обмежує їх застосування в реальному часі.

Оскільки камера може визначити положення об’єкта на її матриці, для визначення місцеположення БПЛА у просторі необхідно використовувати систему відеокамер. Ця система дозволяє отримувати багатовимірні дані про положення об’єкта.

Триангуляція — це геометричний метод, який використовується для визначення координат точки в просторі за допомогою вимірювання кутів відносно відомих точок у просторі.

У контексті визначення місцеположення БПЛА система відеокамер може використовуватися для вимірювання азимутів та кутів місця БПЛА відносно оптичної осі кожного відеодатчика. Ці виміряні кути потім можна використовувати для розрахунку координат БПЛА за допомогою триангуляції.

Використання триангуляції дозволяє отримувати точні координати об'єкта в просторі, що є критично важливим для задач моніторингу.

Важливо усвідомлювати, що будь-яка система або пристрій, що використовується для визначення місцеположення БПЛА, генерує дані з неминучою похибкою. Ці похибки можуть суттєво вплинути на точність розрахунку координат БПЛА, особливо якщо він знаходиться в русі. Неточне визначення координат може призвести до помилкових рішень та небезпечних ситуацій, оскільки БПЛА може бути помилково ідентифікований як розташований в іншому місці. Це ускладнює відстеження руху БПЛА. Це критичний момент для силових структур та військових, яким дуже важливо швидке та точне визначення координат рухомої цілі та її відстеження у просторі в режимі реального часу для швидкого прийняття рішень, наприклад, для знищення неприятельських БПЛА.

Для мінімізації впливу похибок та покращення точності визначення місцеположення БПЛА використовується фільтр Калмана. Фільтр Калмана — це рекурсивний алгоритм траєкторної обробки інформації, який використовує поточні та попередні дані вимірювання для отримання оптимальної оцінки стану динамічної системи. У контексті визначення місцеположення БПЛА фільтр Калмана може використовуватися для об'єднання вимірювань з різних камер та зменшення впливу шуму та похибок на розрахункове фінальне розташування БПЛА у місцевій системі координат. Це дозволяє значно підвищити точність та надійність визначення координат об'єкта, що є критично важливим для успішного виконання задач моніторингу та управління БПЛА.

Висновки

Отже, завдяки швидкому розвитку технологій і зростанню кількості БПЛА, з'являється потреба у створенні ефективних систем для їх виявлення, ідентифікації та відстеження. Таким чином виявлення та відслідковування БПЛА за допомогою даних від мережі відеодатчиків в режимі реального часу є актуальною проблемою у теперішньому часу у військовій і та інших сферах людської діяльності.

Реалізація систем виявлення та відстеження рухомих об’єктів дозволяє оперативно реагувати на можливі загрози, забезпечувати безпеку територій та об'єктів, які перебувають під охороною.

У рамках даної магістерської дисертації будуть розглядатися малі, міні та мікро БПЛА, які відносяться до класу I , відповідно до стандарту НАТО 4670.

2 АНАЛІЗ АЛГОРИТМІВ ВИЯВЛЕННЯ РУХОМИХ ОБ'ЄКТІВ НА ВІДЕО

Оптичний потік

Оптичний потік — це модель видимого руху об’єктів зображення між двома послідовними кадрами, викликаного рухом об’єкта або камери[4]. Це 2D векторне поле, де кожен вектор є вектором зміщення, що показує рух точок від першого кадру до другого. Розглянемо зображення нижче (рис. 2.1).

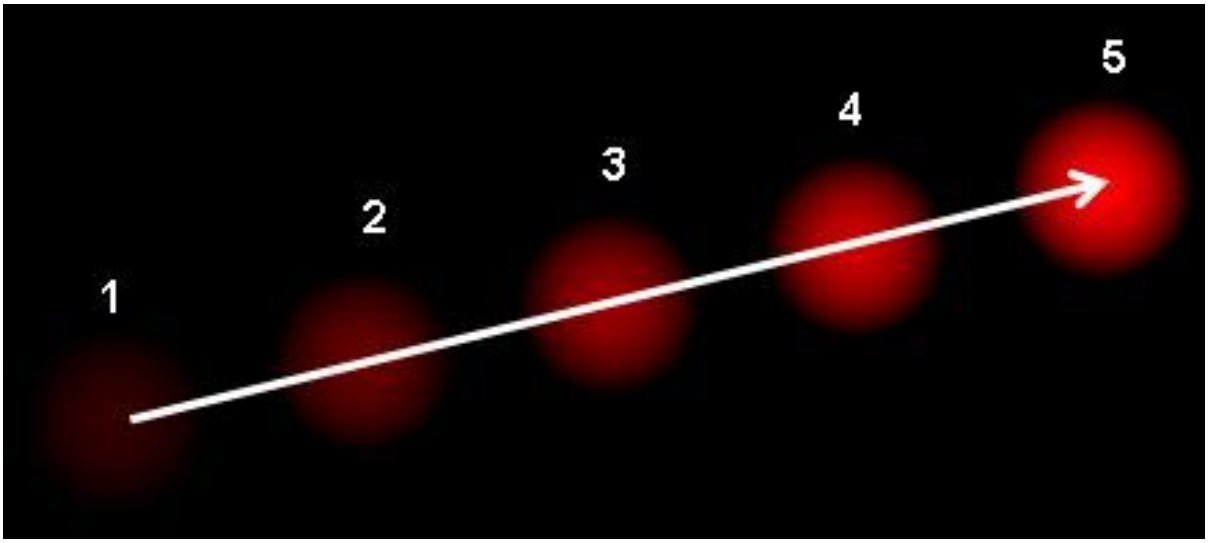


Рисунок 2.1 — Приклад вектору зміщення

Оптичний потік працює на кількох припущеннях:

- Інтенсивність пікселів об’єкта не змінюється між послідовними кадрами.
- Сусідні пікселі мають подібний рух.

Розглянемо піксель *I(x,y,t)* у першому кадрі. Він переміщається на відстань *(dx,dy)* у наступному кадрі, зробленому після часу *dt*. Отже, оскільки ці пікселі однакові, а інтенсивність не змінюється, ми можемо сказати, що

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \qquad (2.1)$$

Якщо взяти апроксимацію ряду Тейлора правої частини, видалити спільні члени та поділити на dt, отримаємо наступне рівняння

$$f_x u + f_y v + f_t = 0 \qquad (2.2)$$

де $f_x = \frac{\partial f}{\partial x}$, $f_y = \frac{\partial f}{\partial y}$, $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$

Наведене вище рівняння називається рівнянням оптичного потоку. У ньому ми можемо знайти fx і fy, це градієнти зображення. Подібним чином ft є градієнтом за часом. Але (u,v) невідомо.

Ми не можемо розв'язати одне рівняння з двома невідомими змінними. Отже, існує кілька методів вирішення цієї проблеми, і один із них — метод Lucas-Kanade.

Раніше було припущення, що всі сусідні пікселі матимуть подібний рух. Метод Лукаса-Канаде використовує невелику ділянку розміром 3 на 3 навколо точки. Отже, усі 9 точок мають однаковий рух. Ми можемо знайти (fx, fy, ft) для цих 9 точок. Отже, тепер нашою проблемою стає розв'язування 9 рівнянь із двома невідомими змінними, які перевизначені. Краще рішення отримано методом найменших квадратів. Нижче наведено остаточне рішення, яке складається з двох рівнянь і двох невідомих задач, які потрібно вирішити, щоб отримати рішення.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \Sigma_i \mathrm{f_{x_i}}^2 & \Sigma_i \mathrm{f_{x_i} f_{y_i}} \\ \Sigma_i \mathrm{f_{x_i} f_{y_i}} & \Sigma_i \mathrm{f_{y_i}}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\Sigma_i \mathrm{f_{x_i} f_{t_i}} \\ -\Sigma_i \mathrm{f_{y_i} f_{t_i}} \end{bmatrix} \qquad (2.3)$$

Приклад результату методу Lucas-Kanade

Рисунок 2.2 — Результат роботи методу Lucas-Kanade

Meanshift

Ідея, яка лежить в алгоритму Meanshift, проста. Припустимо, що у нас є набір точок. Надається невелике вікно, і ми повинні перемістити це вікно в область максимальної щільності пікселів (або максимальної кількості точок)[5]. Це показано на простому зображенні нижче (рис. 2.3)

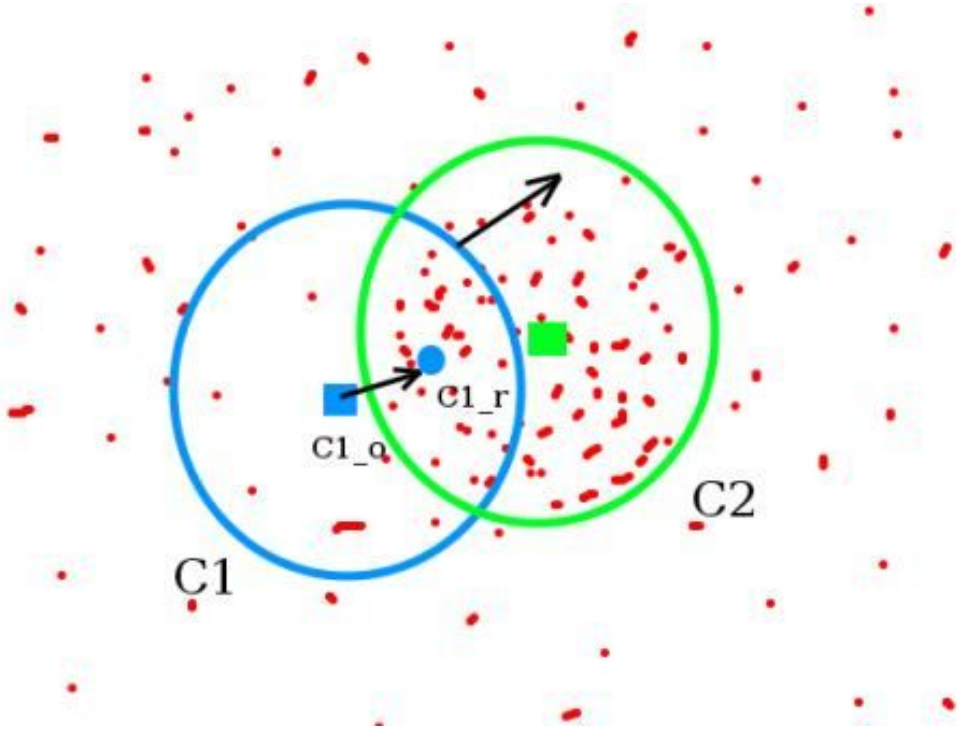


Рисунок 2.3 — Ілюстрація ідеї алгоритму Meanshift

Початкове вікно показано синім колом із назвою «C1». Його початковий центр позначено синім прямокутником під назвою "C1_o". Але якщо ми знайдете центроїду точок у цьому вікні, отримаємо точку «C1_r» (позначену маленьким синім колом), яка є справжньою центроїдою вікна. Звичайно, вони не збігаються. Тому треба перемістити вікно так, щоб коло нового вікна збігалося з центроїдою попереднього. Після цього треба знову знайти нову центроїду.

Якщо вони не збігаються, то треба повторити процес. Тоді такі ітерації будуть продовжуватися, доки центр вікна та центр центроїди не будуть знаходитися в одній точці.

В результаті буде отримане вікно з максимальним розподілом пікселів. Він позначений зеленим колом із назвою «C2». Як видно на зображенні, він має максимальну кількість точок. Зазвичай передається гістограмі зворотне

спроектоване зображення та початкове цільове розташування. Коли об'єкт рухається рух відображається на гістограмі спроектованому зображенні,. У результаті алгоритм Meanshift переміщує вікно на нове місце з максимальною щільністю. Результат роботи даного алгоритму представлений на наступному рисунку (рис2.4).

Рисунок 2.4 — Результат роботи алгоритму Meanshift

CamShift

Meanshift має недолік, який полягає, у тому, що розмір вікна залишається завжди сталим [5]. Це погано, якщо рух об'єкту спрямований до камери або від неї. Внаслідок цього було знайдено рішення у вигляді нового методу Camshift. Він вирішує цей недолік, змініючи розмір вікна під час виконання алгоритму. Результат роботи даного алгоритму представлений на наступному рисунку (рис. 2.5).

Рисунок 2.5 — Результат роботи алгоритму Camshift

Метод міжкадрової різниці

Одним із методів виділення рухомих об’єктів є знаходження різниця між сусідніми кадрами відеопослідовності.

Візьмемо за приклад задачу визначення рухомих об'єктів на статичному фоні. Алгоритми автоматичного виявлення рухомих об'єктів, що ґрунтуються на міжкадровій обробці, зазвичай аналізують не одне, а декілька (не менше двох) зображень, отриманих у різні моменти часу, або, точніше, відповідні їм сигнали

Найбільша проблема при цьому полягає в тому, що рухомі цілі зазвичай менш контрастні, ніж нерухомі предмети в секторі, який бачить відеодатчик. Це ускладнює виділення корисного сигналу від рухомої цілі на тлі заважаючих фонових сигналів за допомогою простої амплітудної вибірки. Тому процес виявлення інформації про рухомі об'єкти поділяється на два етапи:

1) Створення міжкадрового сигналу різниці (МСР), який містить всю інформацію про зміни, що відбуваються в зображенні, та водночас значно зменшує перепади рівня, що відповідають за нерухомі об'єкти у відеокадрі.

2) обробка МСР для виділення необхідних максимально достовірних даних.

Далі будемо вважати, що камера статична на одній точці у просторі. Тоді різниця між сусідніми кадрами буде мати наступний загальний вигляд

$$I_D(x, y) = |I_i(x, y) - I_{i-1}(x, y)| \qquad (2.4)$$

де I_i - теперішній кадр, I_{i-1} - попередній кадр

Приклад різниці зображення знаходиться на рисунку 2.6.

При порівнянні двох кадрів для виявлення рухомих об'єктів можуть виникати інвертовані зображення, тобто копії об'єктів з негативними значеннями пікселів. Це може призвести до помилкового розпізнавання інвертованого зображення як окремого об'єкта, що ускладнює відстеження декількох рухомих об'єктів.

Методи усунення:

- Виділення об’єктів, які темніше фона
- Виділення об’єктів, які світліше фона
- Використовувати додатковий кадр 3 кадр

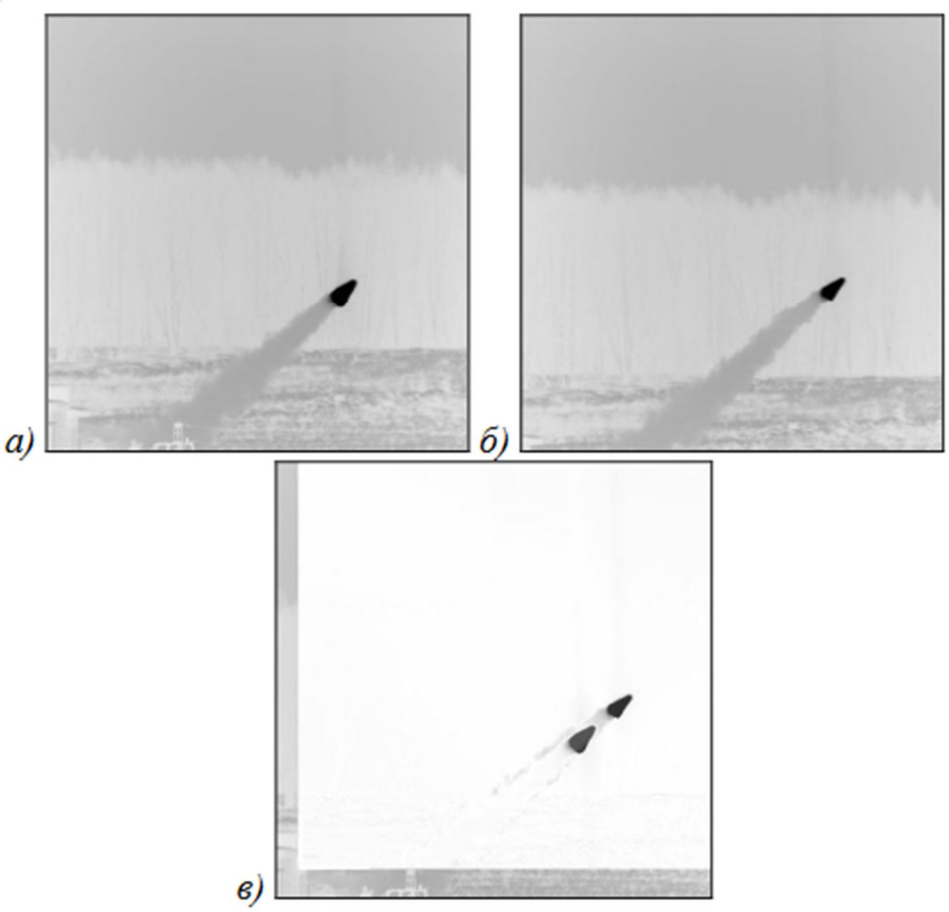


Рисунок 2.6 — Міжкадрові різниця для двох кадрів (до зображення додана інверсія) а) Попередній кадр; б) Теперішній кадр в) Кадрова різниця

Для перших двох варіантів формули мають схожий вигляд. Для виділення об’єктів, які світліше фону формула різниця між сусідніми кадрами буде мати наступний вигляд

$$I_D(x,y) = |I_i(x,y) - I_G(x,y)| \quad (2.5)$$

$$I_G(x,y) = min(I_i(x,y), I_{i-1}(x,y)) \quad (2.6)$$

Для об’єктів, які темніше фону формула різниця між сусідніми кадрами буде мати наступний вигляд

$$I_D(x,y) = |I_i(x,y) - I_G(x,y)|$$

$$I_G(x,y) = max(I_i(x,y), I_{i-1}(x,y)) \quad (2.7)$$

При використанні третього варіанту алгоритм не буде залежати від яскравості об’єктів та фону, але при цьому треба використовувати додатковий третій кадр.

Загальна формула для фінального різнецевого кадру буде мати наступний вигляд:

$$I_D(x,y) = (I_D^1(x,y) - I_D^2(x,y))\theta(I_D^1(x,y) - I_D^2(x,y)) \quad (2.8)$$

де $I_D^1(x,y) = |I_I(x,y) - I_{i-1}(x,y)|$, I_i - теперішній кадр, I_{i-1} - середній кадр

$I_D^2(x,y) = |I_{I-1}(x,y) - I_{i-2}(x,y)|$, I_{i-1} - середній кадр, I_{i-2} - останній кадр

$$\theta(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

Як показали дослідження, у більшості випадків для виділення рухомих об'єктів з МСР достатньо простої бінаризації з використанням глобального порогу. Цей поріг розраховується за наступною формулою:

$$P = \frac{(max(I_D(x,y)) - min(I_D(x,y)))}{2} \quad (2.9)$$

На наступному рисунку (рис. 2.7) зображено теперішній кадр тестового відео.

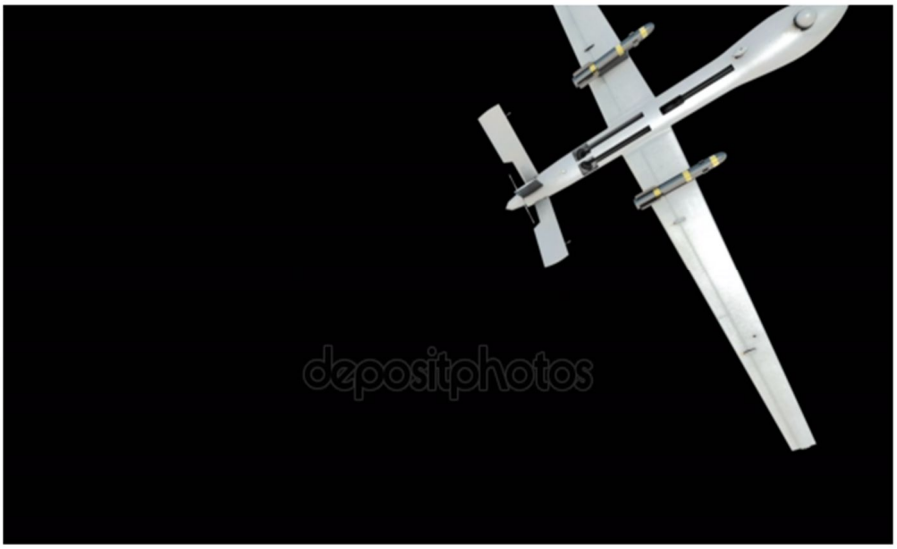


Рисунок 2.7 — Теперішній кадр

На наступному рисунку (рис. 2.8) зображено середній кадр з тестового відео

Рисунок 2.8 — Середній кадр

На рис. 2.9 зображено кадр після середнього, а на рисунку 2.10 зображена різниця між рис. 2.7 і рис. 2.8.

Рисунок 2.9 — Останній кадр

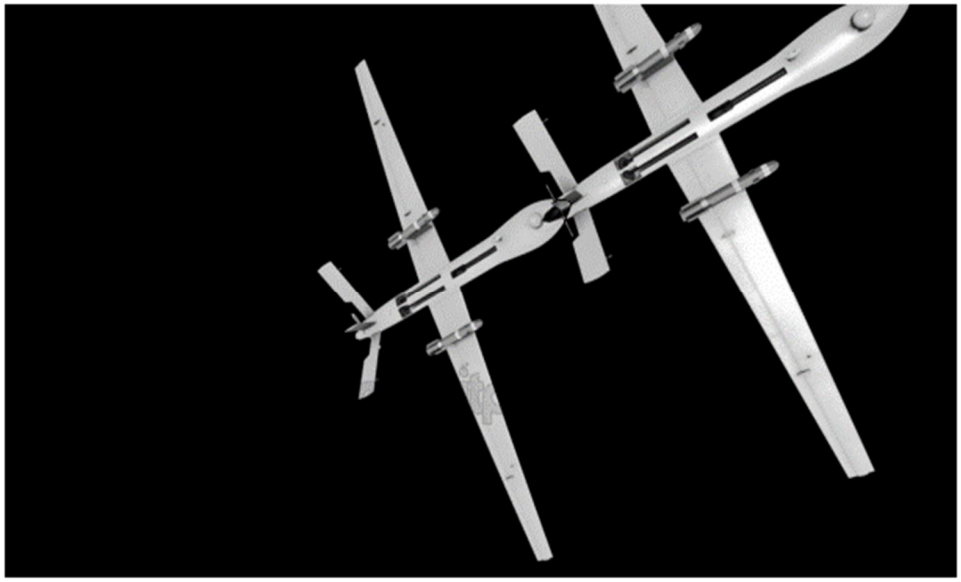


Рисунок 2.10 — Кадрова різниця $I_D^1(x, y)$ до проведення бінарізації

На рис. 2.11 зображена різниця між рис. 2.8 і рис. 2.9. На рис. 2.12 зображено бінарна маска після проведення бінаризації результату роботи алгоритму міжкадрових різниць на основі трьох кадрів.

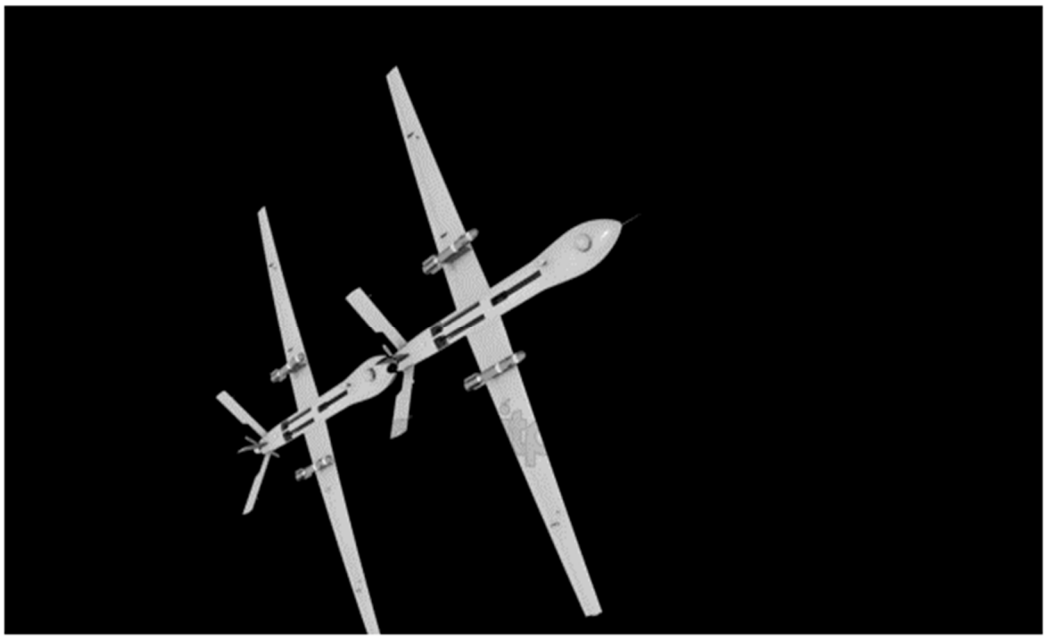


Рисунок 2.11 — Кадрова різниця $I_D^2(x, y)$ до проведення бінарізації

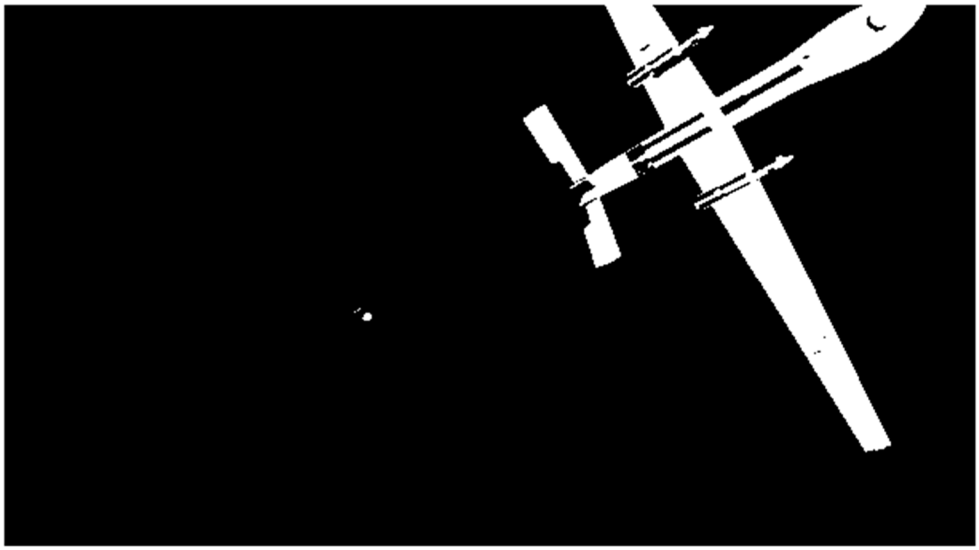


Рисунок 2.12 — Фінальна бінарна маска знайденого об’єкту

Висновки

У цьому розділі порівнюються чотири методи виявлення рухомих об'єктів на відео: метод оптичного потоку, метод Meanshift, метод Camshift та метод міжкадрової різниці.

Метод оптичного потоку:

- Цей метод визначає напрямок та швидкість руху об'єкта на основі зміни яскравості пікселів між кадрами.
- Недоліки: не працює ефективно при зміні освітлення або зміні кольору об'єкта.

Метод Meanshift:

- Більш ефективний при змінному освітленні та зміні кольору об'єкта.
- Порівнює середнє значення пікселів області з цієї ж області на попередньому кадрі.
- Недоліки: може працювати нестабільно при зміні розміру об'єкта, потребує задавати розмір вікна пошуку.

Метод Camshift:

- Вирішує проблему виявлення об'єктів, які змінюють розмір.
- Потребує додаткових обчислювальних потужностей.

- Може працювати не ефективно при зміні освітленості та фону.

Метод міжкадрової різниці:

- Один з найпростіших, швидких та з низькою розрахунковою складністю методів.
- Виділяє рухомі об'єкти шляхом віднімання попереднього кадру від поточного.
- Переваги: досить надійний, працює ефективно при зміні освітлення та кольору об'єкта.
- Недоліки: деякі реалізації можуть показувати інверсні зображення як самостійні об'єкти.

Отже, через високу швидкодію та простоту реалізації, у наступних розділах для розробки алгоритму виявлення та відстеження рухомого об'єкта на основі даних мережі відеокамер буде використовуватися метод міжкадрової різниці.

Для процесу моделювання виявлення рухомого об’єкту на відеопослідовностях буде використовуватися мова програмування Python разом з бібліотеками OpenCV (для зручної роботи з фото та відео файлами) та Numpy (для зручної роботи та прискорення швидкодії при роботі з матрицями на мові Python).

3 РОЗРОБКА АЛГОРИТМУ ВИЯВЛЕННЯ РУХОМОГО ОБ'ЄКТУ НА ВІДЕО З ВИКОРИСТАННЯМ МІЖКАДРОВИХ РІЗНИЦЬ

Структурна схема алгоритму виявлення на відеопослідовностях зображена на рис. 3.1.

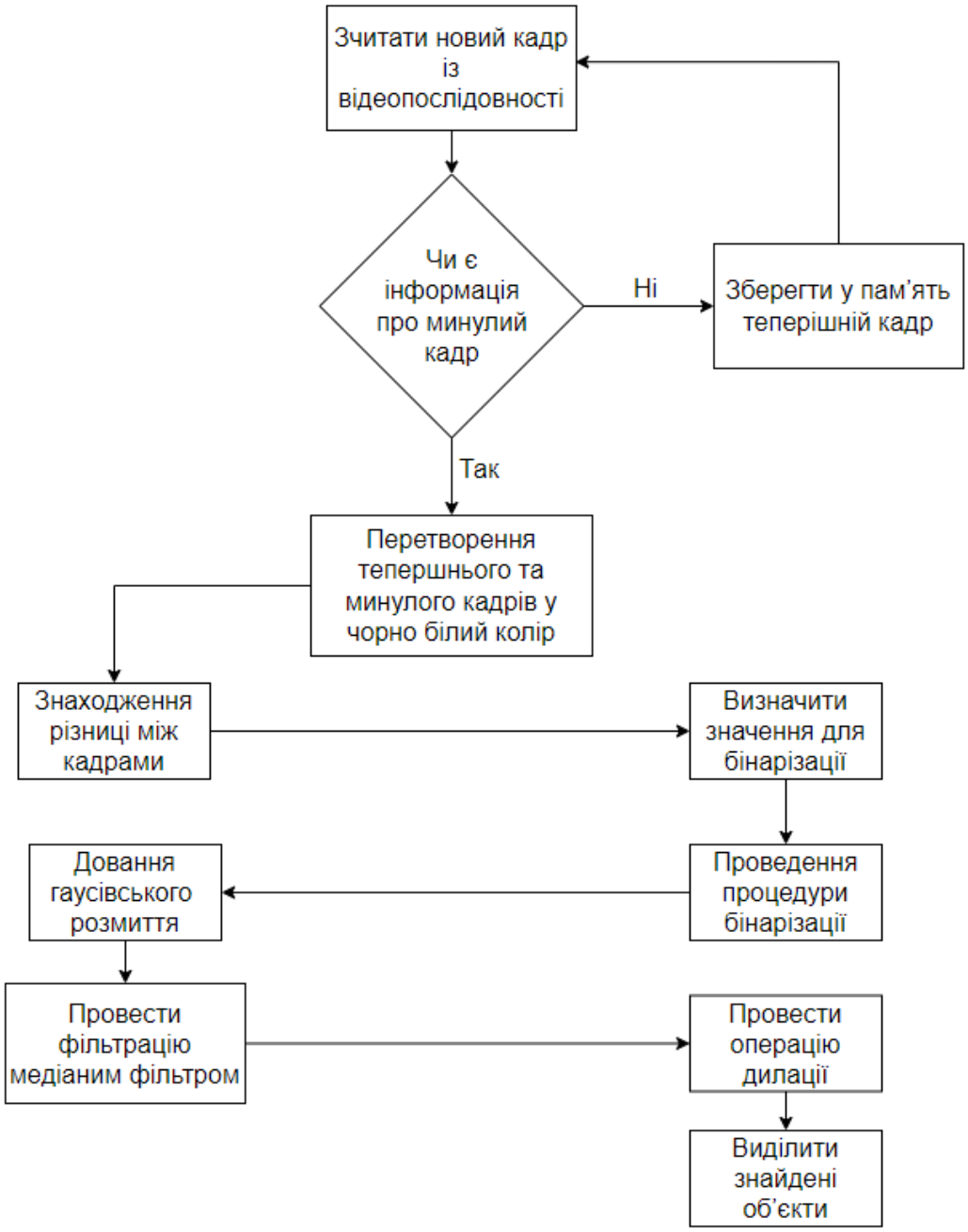


Рисунок 3.1 — Структурна схема алгоритму виявлення рухомого об'єкту

Алгоритм виявлення рухомого об’єкту складається з наступних процедур

Зчитування кадрів

Програма за допомогою бібліотеки OpenCV зчитує кадри з відеопотоку або відеофайлів. Програма не залежить від параметрів самого відео, як, наприклад, ширина, висота кадру, кількість кадрів за секунду, розширення файлу. Програма у кінці циклу помічає опрацьований кадр як минулий і зберігає в оперативну пам’ять ЕОМ для майбутнього порівняння з наступним кадром.

Перевірка прочитання першого кадру

Відбувається перевірка, чи це самий перший кадр був прочитаний. Якщо так, то перший кадр помічається як минулий, зберігається в оперативну пам’ять ЕОМ і програма починає оброблювати другий кадр з файлу. В іншому випадку, робота програми буде аналогічною як описано у пункті вище.

Перетворення збережених у пам’яті кадрів з кольорових у чорно білі

Минулий та теперішній кадри перетворюються з кольорових (програма використовує кольорову модель RGB, але можна використовувати інші моделі за необхідністю) у чорно білі (значення кожного пікселя становить від 0 до 255). За допомогою цього перетворення підвищується точність знаходження контурів рухомої цілі.

Знаходження різниці між кадрами

На цьому кроці відбувається знаходження різниці між кадрами і потім отримання модуля від отриманого результату за допомогою можливостей бібліотеки Numpy для підвищення швидкості роботи алгоритму через наявність вже заздалегідь скомпільованих файлів даної бібліотеки.

Бінаризація міжкадрової різниці

Для більш чіткого виділення контурів рухомих об’єктів та відсікання ділянок, які були помилково розпізнані як рухомий об’єкт проводиться бінарізація, в

результаті якої усі пікселі, які мають значення вище заданого порогу, стають рівними максимальному значенню, а саме 255. У протилежному випадку решта пікселів будуть мати значення, яка дорівнює нулю.

Додавання гаусівського розмиття

Для кращого виділення контуру знайденого об’єкту відбувається додавання гаусівського розмиття

Фільтрація шумів

Для усунення імпульсних шумових складових на бінарій масці, які можуть бути розпізнаними як самостійні об’єкти, виконується фільтрація за допомогою медіаного фільтра.

Проведення дилації

Передостаннім кроком виконується базова математична морфологічна операція дилації. Дана операція має наступну формулу [6]:

$$A \oplus B = \{z \in E | (B^s)_z \cap A \neq \emptyset\} \quad (3.1)$$

Приклади зображення до та після проведення цієї операції зображений на рис. 3.2 та рис. 3.3.

Рисунок 3.2 — Базове зображення

Рисунок 3.3 — Зображення після проведення дилації

Дана морфологічна операція здійснюється для того, щоб об’єднати частини об’єкту в один великий об’єкт на бінарній масці, у випадку якщо виявлена рухома ціль була розпізнана як група маленьких об’єктів.

Виділення контуру знайденої цілі

Після цих кроків виконується виділення контуру знайденої цілі за допомогою функції findContours бібліотеки OpenCV. Ця функція використовує алгоритм пошуку контурів, який розробили Сатоші Сузукі та Кеічі Абе. Для збільшення швидкості роботи та зменьшення кількості спожитої пам’яті ЕОМ використовується метод апроксимації координат контуру. Порівняння використання методу апроксимації зображений на рисунку. Синім кольором виділені усі точки контура, які зберігаються. На лівій фігурі даний метод не використовується. Він застосовується лише для правої фігури.

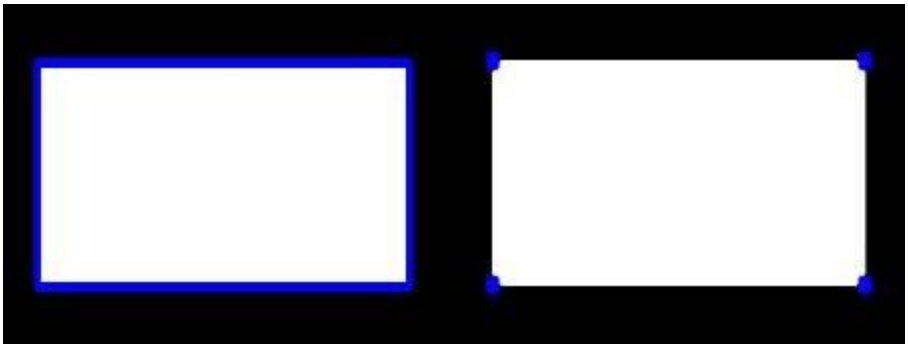


Рисунок 3.4 — Порівняння відображенння координат країв контуру за допомогою різних методів.

Як видно на рисунку, використання апроксимації економить велику кількість пам’яті.

Момент являє собою сумарну характеристику контуру, яка була отримана в результаті суми усіх пікселей[7].

Потім, коли контури були знайдені, відбувається визначення центру маси виявленої цілі за допомогою моментів, які визначаються формулою[7]:

$$M_{i,i} = \sum_x \ \sum_y \ x^i y^j I(x,y) \tag{3.2}$$

де $I(x,y)$ — значення пікселю на зображенні з координатами x, y.

Тоді координати центра маси знаходяться за формулами (3.3), (3.4)[7]:

$$cX = \frac{M_{1,0}}{M_{0,0}} \tag{3.3}$$

$$cY = \frac{M_{0,1}}{M_{0,0}} \tag{3.4}$$

де cX — координата центру маси вздовж осі X;
cY — координати центру маси вздовж осі Y

Аналіз результатів виявлення рухомого об’єкту

Результати даного алгоритму були промодельовані на електронно обчислюванні машині на основі відео, яке було знято на території КПІ, з літаючим БПЛА.

Результати моделювання на основі реального відео зображено на рис. 3.5. Синім кольором позначений шлях, який пройшов рухомий об’єкт, для перевірки на правильність роботи алгоритму. Зеленим кольором позначений строб виявленої рухомої цілі. Червоним кольором позначено ідентифікатор знайденого об’єкта. Індексація починається з нуля.

Рисунок 3.5 — Результат роботи алгоритму виявлення рухомого об’єкта

Рисунок 3.6 — Отримана бінарна маска внаслідок роботи алгоритму

Висновки

У даному розділі був розроблений алгоритм виявлення рухомого об’єкта на відео. В результаті роботи отриманні результати показують, що даний алгоритм працює як і очікується. Допоміжна синя крива шляху рухомої цілі має таку форму шляху БПЛА на відео. Різку зміну центра маси знайденого об’єкта, яка спостерігається, можна пояснити як обрання неоптимального ліміту для бінаризації, так і якістю самого відео, на основі якого відбувалося моделювання.

4 РОЗРОБКА АЛГОРИТМУ ВИЗНАЧЕННЯ КООРДИНАТ РУХОМОГО ОБ'ЄКТУ ЗА ДАНИМИ ДВОХ ВІДЕОКАМЕР У ПРЯМОКУТНІЙ СИСТЕМІ КООРДИНАТ

При постанові задачі визначення координат об'єкта однієї відеокамери недостатньо, бо за її допомогою можна дізнатися лише положення об'єкту на матриці самої камери. Тому для процесу визначення координат треба використовувати систему відеокамер. В даній магістерській дисертації буде розроблений алгоритм визначення координат за допомогою системи із двох відеокамер, які розташовані на певній фіксованій відстані відносно один одної. Оптичні вісі камер паралельні відносно один одної. Взаємне розташування камер у місцевій системі координат показано на рис. 4.1. Усі подальші моделювання будуть проводитися за допомогою програмного забезпечення Matlab.

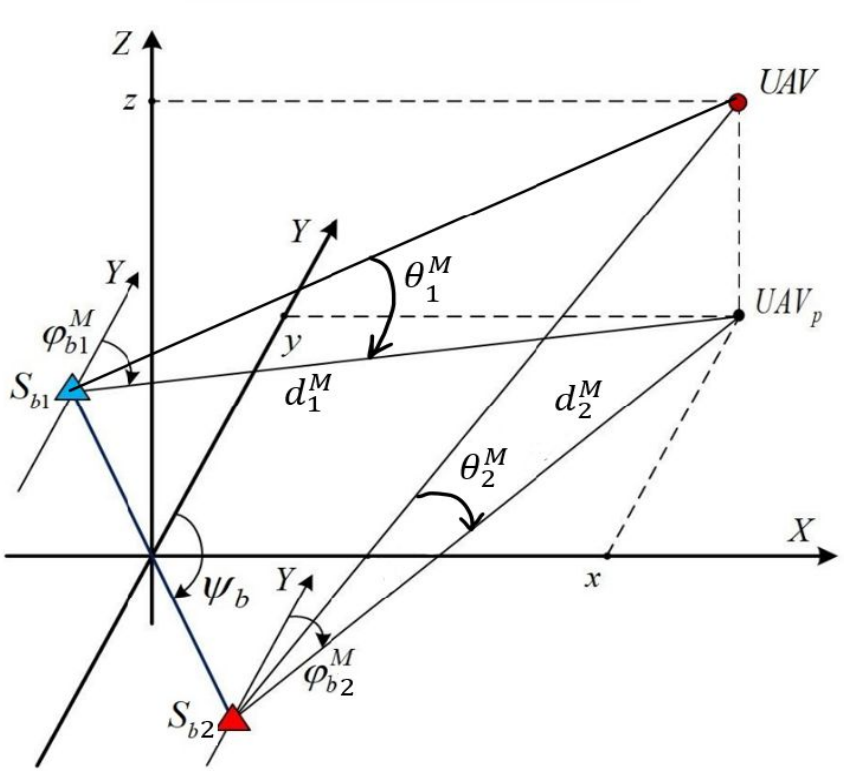


Рисунок 4.1 — Зображення розташування камер та БПЛА під час моделювання

Як правило, відстань між відеокамерою та БПЛА значно перевищує фокусну відстань оптичної системи. Тому будемо використовувати проективну модель відеокамери, в якій проектування зображення тривимірного об'єкта у фокальну площину (площину зображення) виконується через оптичний центр.

Як зазначалося раніше, відеодатчик (на рис. 4.1 — s_{b1} і s_{b2}) може дати лише положення на матриці, але знаючи параметри цього датчика, можна визначити як азимут, так і кут місця. За допомогою алгоритму, який був представлений у попередньому розділі, можна отримати положення центра мас рухомої цілі у системі координат матриці. Цю систему позначимо як UOV, де OU — горизонтальна складова, а OV — вертикальна складова. Знаючи параметри камери, можна зробити перехід з системи координат матриці у систему координат самої камери. Початок системи UOV з'єднаний з верхнім лівим пікселем. Координати центрального пікселя (cx, cy) (оптичний центр) знаходяться на перетині головної оптичної осі об'єктива з матрицею камери. Вони також як і координати будь якої іншої точки на зображенні визначаються в пікселях — округлюються с точністю до пікселя.

На рис. 4.2 зображена ілюстрація розташувань систем координат матриці камери та системи координат відеокамери. Точка P на даному рисунку являє собою

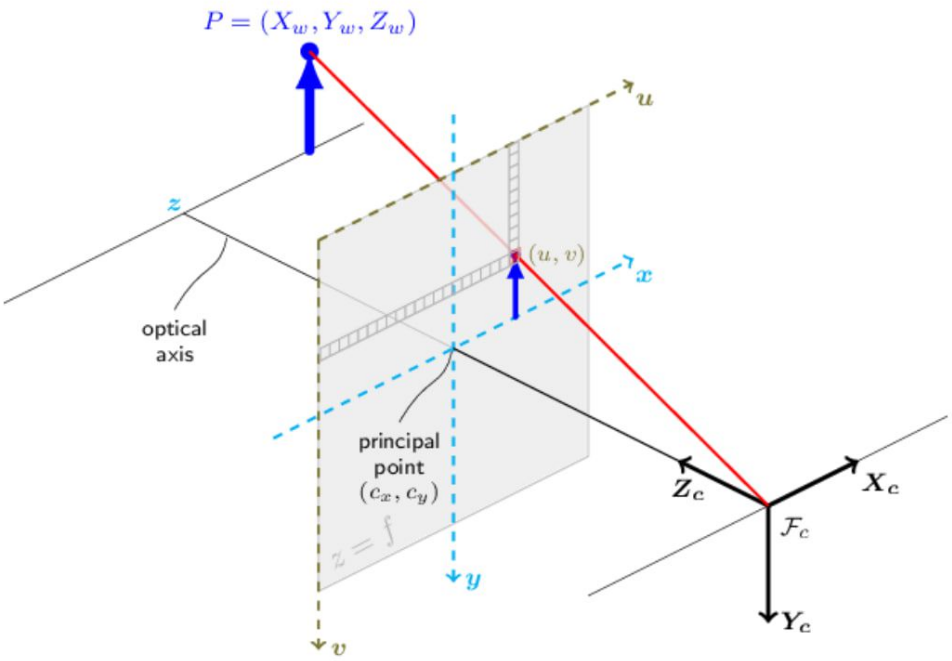


Рисунок 4.2 — Ілюстрація розташування систем координат матриці камери та системи координат відеокамери

Для знаходження азимута та кута місця треба визначити кути відхилу по горищонталі та вертикалі відносно системи координат відеокамери. Спочатку

треба зробити перехід від системи координат матриці до системи координат відеодатчика.

Зворотнє перетворення робиться за формулою (4.1)[8]

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y_c \\ Z_c \end{bmatrix}, \quad (4.1)$$

де X_c, Y_c, Z_c — положення точки в системі координат камери,

u, v — положення точки в системі координат матриці після перетворення,

c_x — значення половини ширини матриці у пікселях,

c_y — значення половини висоти матриці у пікселях,

$$f_x = {}^{f}/_{w} \quad (4.2)$$

де f — фокальна довжика камери,

w — ширина матриці у пікселях.

$$f_y = {}^{f}/_{h}, \quad (4.3)$$

де h — висота матриці у пікселях.

Використовуючи (4.1), отримуємо формули для перерахунку координат точки, яка знаходиться на матриці камери, у систему координат відеодатчика.

$$\Delta X = \frac{(u-u_0)w}{f} Z \quad (4.4)$$

$$\Delta Y = \frac{(v-v_0)h}{f} Z \quad (4.5)$$

У випадку, коли розраховане положення виявленої цілі було отримано не у місцевій системі координат, то слід використати наступні формули для переходу до місцевої системи координат за допомого повороту на певний кут відносно осі[9].
Поворот відносно осі OX на кут α

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \quad (4.6)$$

Поворот відносно осі OY на кут β

$$R_y(\beta) = \begin{bmatrix} cos\,\beta & 0 & sin\,\beta \\ 0 & 1 & 0 \\ -sin\,\beta & 0 & cos\,\beta \end{bmatrix} \quad (4.7)$$

Поворот відносно осі OZ на кут γ

$$R_z(\gamma) = \begin{bmatrix} cos\,\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.8)$$

У випадку, коли треба робити поворот для усіх трьох координат, то слід перемножити між собою матриці (4.6), (4.7) та (4.8). В результаті отримуємо наступну матрицю

$$R(\alpha,\beta,\gamma) = R_z(\gamma)R_y(\beta)R_x(\alpha) =$$

$$\begin{bmatrix} cos\,\alpha\,cos\,\beta & -\sin\alpha\ \sin\alpha\ \sin\alpha - \ \sin\alpha\ \sin\alpha & cos\,\alpha\,sin\,\beta\,cos\,\gamma + \ sin\,\alpha\,sin\,\gamma \\ \sin\alpha\ \ cos\,\beta & \cos\alpha & sin\,\alpha\,sin\,\beta\,cos\,\gamma + \ cos\,\alpha\sin\gamma \\ -sin\,\beta & cos\,\beta\sin\gamma & cos\,\beta\,cos\,\gamma \end{bmatrix}$$

Для зручності, позначимо отриманої матриці як R_{ij},

де i = 1, 2, 3 – номер рядка матриці

j = 1, 2, 3 – номер стовпця матриці

$$R(\alpha,\beta,\gamma) = R_z(\gamma)R_y(\beta)R_x(\alpha) = \begin{bmatrix} R_{11} & R_{21} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad (4.9)$$

Кути α, β, γ слід визначати за правилом правої руки, як показано на рис. 4.3.

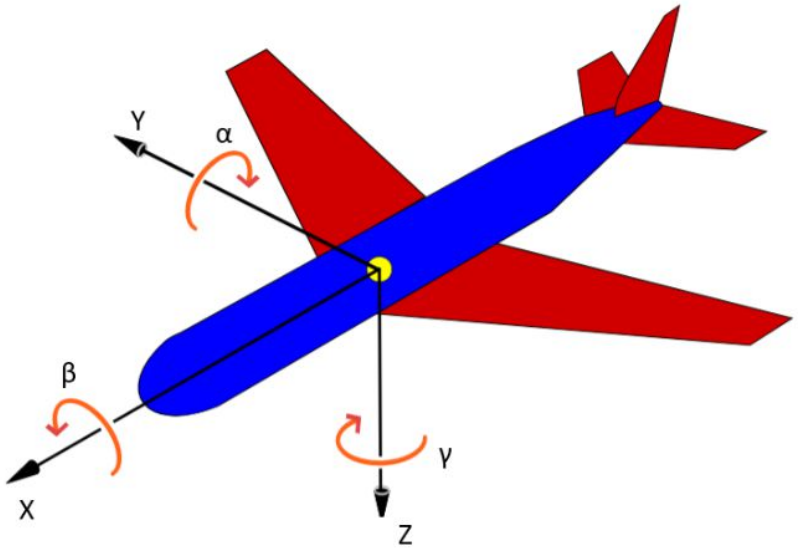


Рисунок 4.3 — Ілюстрація вибору кута повороту для кожної координатної осі

Якщо крім повороту треба робити переміщення вздовж координат, то слід розширити матрицю (4.9), додавши новий стовпець, який буде містити вектор зміщення, який буде містити значення T_x для зміщення відносно осі X, T_y — відносно осі Y і T_z — відносно осі Z. В результаті маємо наступну формулу для здійснення переходу з однієї системи координат в іншу:

$$\begin{bmatrix} X_N \\ Y_N \\ Z_N \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (4.10)$$

Знаючи значення фокальної довжини відеокамери можна розрахувати азимут та кут місця для подальшого визначення координат рухомої цілі.

$$\varphi_{b1,2}^{M} = \pi - ({}^{\pi}/_{2} - \operatorname{atan}(\tan(\tfrac{\Delta X}{f}))) \quad (4.11)$$

$$\theta_{b1,2}^{M} = {}^{\pi}/_{2} - \operatorname{atan}(\tan(\tfrac{\Delta Y}{f})) \quad (4.12)$$

Після розглянутих кроків, за допомогою теореми синусів можна визначити відстань від датчика s_{b1} до проекції цілі на горизонтальну площину XOY місцевої системи координат рис. 4.1

$$d_1^M = \frac{b \cdot sin(180° - (\psi_b - \varphi_2^M))}{sin((\psi_b - \varphi_2^M) - (\psi_b - \varphi_1^M))} \quad (4.13)$$

де b — база системи з двох відеокамер (відстань між двома відеокамерами)

$$b = \sqrt{(x_{b2} - x_{b1})^2 + (y_{b2} - y_{b1})^2}, \quad (4.14)$$

де ψ_b — .пеленг бази.

Враховуючи розраховані кути мережі відеодатчиків отримуємо значення координат БПЛА в місцевій прямокутній системі координат рис. 4.1 за формулами:

$$x^M = d_1^M \sin \varphi_1^M \quad (4.15)$$

$$y^M = d_1^M \cos \varphi_1^M \quad (4.16)$$

$$z^M = d_1^M tg\theta_1^M \quad (4.17)$$

З урахуванням похибки вимірювання перепишемо вирази для кутів, які вимірюються:

$$\varphi_1^M = \varphi_1 + \Delta\varphi_1 \quad (4.18)$$

$$\varphi_2^M = \varphi_2 + \Delta\varphi_2 \quad (4.19)$$

$$\theta_1^M = \theta_1 + \Delta\theta_1 \quad (4.20)$$

де φ_1^M, φ_2^M, θ_1^M — виміряні значення кутів, φ_1, φ_2, θ_1 — істині значення кутів, $\Delta\varphi_1$, $\Delta\varphi_2$, $\Delta\theta_1$ — похибки вимірювання кутів.

Підставимо (4.18), (4.19) в формулу (4.13). Зробимо розклад нелінійних функцій в ряд Тейлора і обмежуючись членами першого порядку отримаємо наступні вирази:

$$\begin{gathered} sin\big((\psi_b - \varphi_2^M) - (\psi_b - \varphi_1^M)\big) = sin(\varphi_1^M - \varphi_2^M) = \\ = sin(\varphi_1 - \varphi_2 + \Delta\varphi_1 - \Delta\varphi_2) \approx sin(\varphi_1 - \varphi_2) + \\ +(\Delta\varphi_1 - \Delta\varphi_2)\, cos(\varphi_1 - \varphi_2) \end{gathered} \quad (4.21)$$

$$\begin{gathered} sin\big(180° - (\psi_b - \varphi_2^M)\big) \approx \\ \approx sin\big(180° - (\psi_b - \varphi_2)\big) + \\ + \Delta\varphi_2\, cos(180° - (\psi_b - \varphi_2)); \end{gathered} \quad (4.22)$$

Підставимо вирази (4.22) в формулу (4.13)

$$d_1^M = \frac{b \cdot (sin(180^\circ - (\psi_b - \varphi_2)) + \Delta\varphi_2\, cos(180^\circ - (\psi_b - \varphi_2)))}{sin(\varphi_1 - \varphi_2) + (\Delta\varphi_1 - \Delta\varphi_2)\, cos(\varphi_1 - \varphi_2)} \approx$$

$$\approx \left| \frac{1}{1+x} \approx 1 - x \right| \approx$$

$$\approx \frac{b \cdot (sin(180^\circ - (\psi_b - \varphi_2)) + \Delta\varphi_2\, cos(180^\circ - (\psi_b - \varphi_2)))}{sin(\varphi_1 - \varphi_2)} \times$$

$$\times \left[1 - (\Delta\varphi_1 - \Delta\varphi_2) \frac{cos(\varphi_1 - \varphi_2)}{sin(\varphi_1 - \varphi_2)}\right] = \quad (4.23)$$

$$= \frac{b \cdot sin(180^\circ - (\psi_b - \varphi_2))}{sin(\varphi_1 - \varphi_2)} + \frac{b \cdot \Delta\varphi_2\, cos(180^\circ - (\psi_b - \varphi_2))}{sin(\varphi_1 - \varphi_2)} -$$

$$- \frac{b \cdot sin(180^\circ - (\psi_b - \varphi_2))\, cos(\varphi_1 - \varphi_2)}{(sin(\varphi_1 - \varphi_2))^2} (\Delta\varphi_1 - \Delta\varphi_2)$$

$$= d_1 + \Delta d_1$$

Отримуємо вираз для помилки вимірювання по дальності d_1^M:

$$\Delta d_1 = \frac{b \cdot cos(180^\circ - (\psi_b - \varphi_2))}{sin(\varphi_1 - \varphi_2)} \Delta\varphi_2 -$$

$$- \frac{b \cdot sin(180^\circ - (\psi_b - \varphi_2))\, cos(\varphi_1 - \varphi_2)}{(sin(\varphi_1 - \varphi_2))^2} (\Delta\varphi_1 - \Delta\varphi_2) =$$

$$= \left(- \frac{b \cdot sin(180^\circ - (\psi_b - \varphi_2))\, cos(\varphi_1 - \varphi_2)}{(sin(\varphi_1 - \varphi_2))^2}\right) \Delta\varphi_1 + \quad (4.24)$$

$$+ \frac{b}{sin(\varphi_1 - \varphi_2)} (cos(180^\circ - (\psi_b - \varphi_2)) -$$

$$- \frac{sin(180^\circ - (\psi_b - \varphi_2))\, cos(\varphi_1 - \varphi_2)}{sin(\varphi_1 - \varphi_2)}) \Delta\varphi_2$$

Отже, помилку Δd_1 можна записати наступним чином:

$$\Delta d_1 = c_1 \cdot \Delta\varphi_1 + c_2 \cdot \Delta\varphi_2 \quad (4.25)$$

де c_1, c_2 – коефіцієнти, що визначаються за формулами:

$$c_1 = - \frac{b \cdot sin(180^\circ - (\psi_b - \varphi_2))\, cos(\varphi_1 - \varphi_2)}{(sin(\varphi_1 - \varphi_2))^2} \quad (4.26)$$

$$c_2 = \frac{b}{sin(\varphi_1 - \varphi_2)} \left(cos(180^\circ - (\psi_b - \varphi_2)) - \frac{sin(180^\circ - (\psi_b - \varphi_2))\, cos(\varphi_1 - \varphi_2)}{sin(\varphi_1 - \varphi_2)}\right) \quad (4.27)$$

Використавши формули (4.18)-(4.20), отримуємо місцеположення цілі в місцевій системі координат. Виміряне значення координати x^M можна представити у вигляді

$$x^M = (d_1 + \Delta d_1) \cdot sin(\varphi_1 + \Delta\varphi_1) = \\ =d_1 \sin\varphi_1 + \Delta d_1 \sin\varphi_1 + \\ +\Delta\varphi_1 d_1 \cos\varphi_1 \tag{4.28}$$

З використанням (4.25) отримуємо

$$x^M = d_1 \sin\varphi_1 + \Delta\varphi_2 c_1 \sin\varphi_1 + \\ \Delta\varphi_1(c_1 \cos\varphi_1 + d_1 \cos\varphi_1) = x + \Delta x \tag{4.29}$$

Таким чином помилка вимірювання Δx розраховується за формулою

$$\Delta x = \beta_1 \cdot \Delta\varphi_1 + \beta_2 \cdot \Delta\varphi_2 \tag{4.30}$$

де β_1, β_2 – коефіцієнти, що розраховуються за формулами:

$$\beta_1 = c_1 \cos\varphi_1 + d_1 \cos\varphi_1 \tag{4.31}$$

$$\beta_2 = c_1 \sin\varphi_1 \tag{4.32}$$

Значення координати y^M можна переписати наступним чином

$$y^M = (d_1 + \Delta d_1) \cdot cos(\varphi_1 + \Delta\varphi_1) = d_1 \cos\varphi_1 + \Delta d_1 \cos\varphi_1 - \\ -\Delta\varphi_1 d_1 \sin\varphi_1$$

З використанням (43.25) отримуємо

$$y^в = d_1 \cos\varphi_1 + \Delta\varphi_2 c_2 \cos\varphi_1 + \\ +\Delta\varphi_1(c_1 \cos\varphi_1 - d_1 \sin\varphi_1) = y + \Delta y \tag{4.33}$$

Отже, помилка вимірювання Δy розраховується за формулою

$$\Delta y = \alpha_1 \cdot \Delta\varphi_1 + \alpha_2 \cdot \Delta\varphi_2 \quad (4.34)$$

де α_1, α_2 – коефіцієнти, що розраховуються за формулами:

$$\alpha_1 = c_1 \cos\varphi_1 - d_1 \sin\varphi_1; \quad (4.35)$$

$$\alpha_2 = c_2 \cos\varphi_1 \quad (4.36)$$

Значення координати y^M можна переписати наступним чином

$$z^M = (d_1 + \Delta d_1)tg(\theta_1 + \Delta\theta) = d_1 tg\theta_1 + \Delta d_1 tg\theta_1 + \frac{d_1}{\cos^2\theta_1}\Delta\theta \quad (4.37)$$

З використанням (4.25) отримуємо

$$z^M = d_1 tg\theta_1 + c_1 tg\theta_1 \Delta\varphi_1 + c_2 tg\theta_1 \Delta\varphi_2 + \frac{d_1}{\cos^2\theta_1}\Delta\theta_1 \quad (4.38)$$

Таким чином помилка вимірювання Δz розраховується за формулою

$$\Delta z = \gamma_1 \cdot \Delta\varphi_1 + \gamma_2 \cdot \Delta\varphi_2 + \gamma_3 \Delta\theta_1 \quad (4.39)$$

де $\gamma_1, \gamma_2, \gamma_3$ – коефіцієнти, що розраховуються за формулами:

$$\gamma_1 = c_1 tg\theta_1 \quad (4.40)$$

$$\gamma_2 = c_2 tg\theta_1 \quad (4.41)$$

$$\gamma_3 = \frac{d_1}{\cos^2\theta_1}, \quad (4.42)$$

За допомогою формул (4.25), (4.30), (4.34), (4.39) отримуємо дисперсії помилок визначення координат цілі, а також дальності d_1

$$\sigma_x^2 = \beta_1^2 \cdot \sigma_{\varphi_1}^2 + \beta_2^2 \cdot \sigma_{\varphi_2}^2 \quad (4.43)$$

$$\sigma_y^2 = \alpha_1^2 \cdot \sigma_{\varphi_1}^2 + \alpha_2^2 \cdot \sigma_{\varphi_2}^2 \quad (4.44)$$

$$\sigma_z^2 = \gamma_1^2 \cdot \sigma_{\varphi_1}^2 + \gamma_2^2 \cdot \sigma_{\varphi_2}^2 + \gamma_3^2 \cdot \sigma_{\theta_1}^2 \quad (4.45)$$

$$\sigma_{d_1}^2 = c_1^2 \cdot \sigma_{\varphi_1}^2 + c_2^2 \cdot \sigma_{\varphi_2}^2 \quad (4.46)$$

Взаємна кореляція між помилкам визначається за формулами

$$\sigma_{xy} = \alpha_1 \cdot \beta_1 \cdot \sigma_{\varphi_1}^2 + \alpha_2 \cdot \beta_2 \cdot \sigma_{\varphi_2}^2 \quad (4.47)$$

$$\sigma_{xz} = \alpha_1 \cdot \gamma_1 \cdot \sigma_{\varphi_1}^2 + \alpha_2 \cdot \gamma_2 \cdot \sigma_{\varphi_2}^2 \quad (4.48)$$

$$\sigma_{yz} = \beta_1 \cdot \gamma_1 \cdot \sigma_{\varphi_1}^2 + \beta_2 \cdot \gamma_2 \cdot \sigma_{\varphi_2}^2 \quad (4.49)$$

Аналіз точносних характеристик розробка алгоритму визначення координат рухомого об’єкту за даними двох відеокамер проведено шляхом статистичного моделювання на ЕОМ. Розгляд результатів точносних характеристик, які були отримані під час моделювання у Matlab, виконаний для випадку вимірювання координат цілі в просторі. Дистанція між відеодатчиками дорівнює b =100 м., ψ_b =90 град. Ціль рухається навколо відеодатчика 1 по траєкторії радіусом 1000 м і на висоті 100 м. На рис. 4.4 наведено істинну і виміряну траєкторії руху цілі на площині XY.

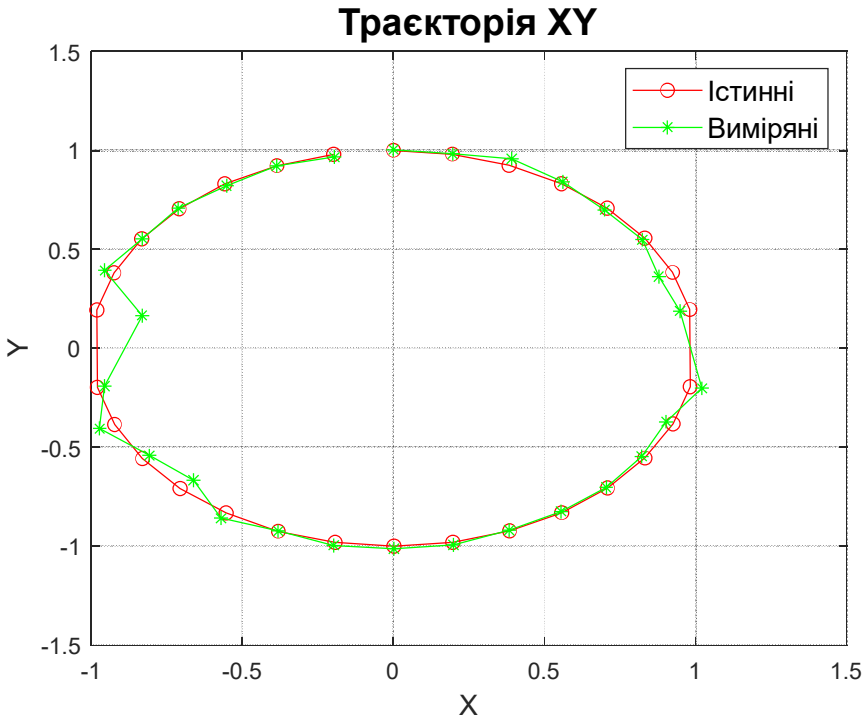


Рисунок 4.4 — Істинна і виміряна траєкторії руху цілі.

Розглянуто точносні характеристики визначення координат для різних траєкторій руху. На рис. 4.5 зображені графіки залежності СКВ помилок вимірювань координат цілі та СКВ помилки вимірювання дальності від кута приходу сигналу з радіусом траєкторії 1000 метрів; на рис. 4.6 – з радіусом 600 метрів; на рис. 4.7 – з радіусом 200 метрів. На рис. 4.8 показано залежності взаємна кореляція помилок визначення координат цілі від кута приходу сигналу з радіусом траєкторії 1000 метрів.

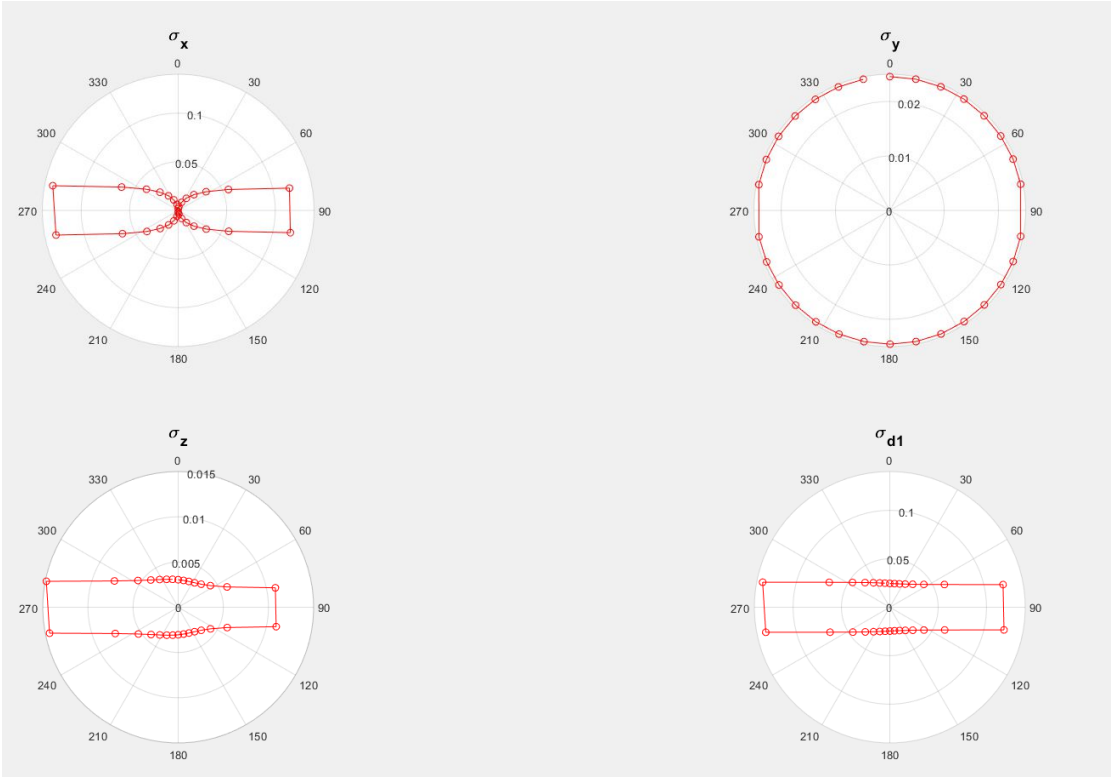


Рисунок 4.5 —Залежності СКВ помилок вимірювань координат цілі від кута приходу сигналу з радіусом траєкторії 1000 метрів

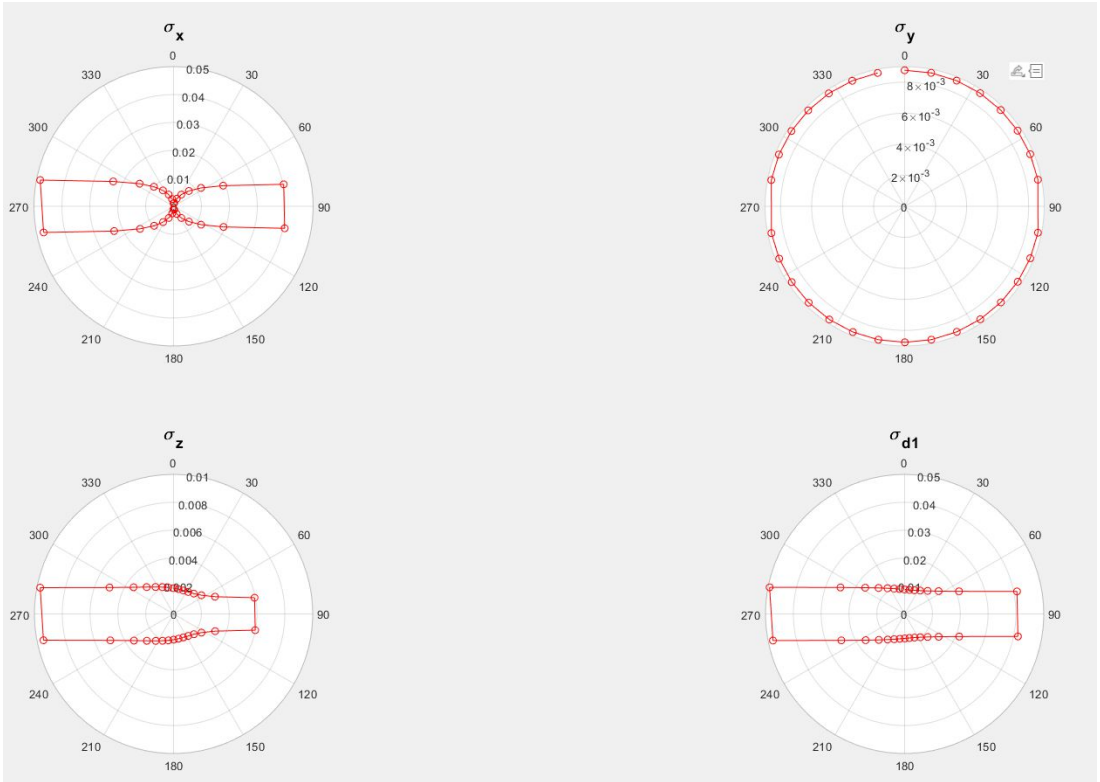


Рисунок 4.6 — Залежності СКВ помилок вимірювань координат цілі від кута приходу сигналу з радіусом траєкторії 600 метрів

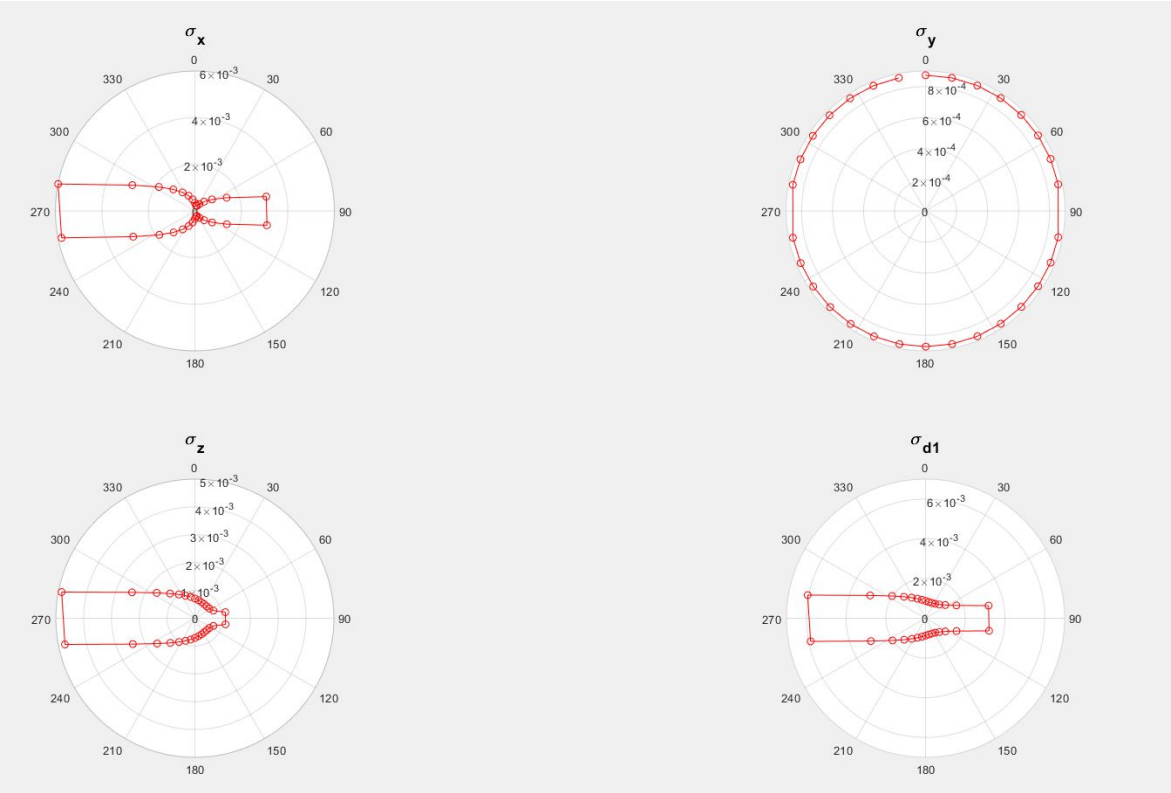


Рисунок 4.7 — Залежності СКВ помилок вимірювань координат цілі від кута приходу сигналу з радіусом траєкторії 200 метрів

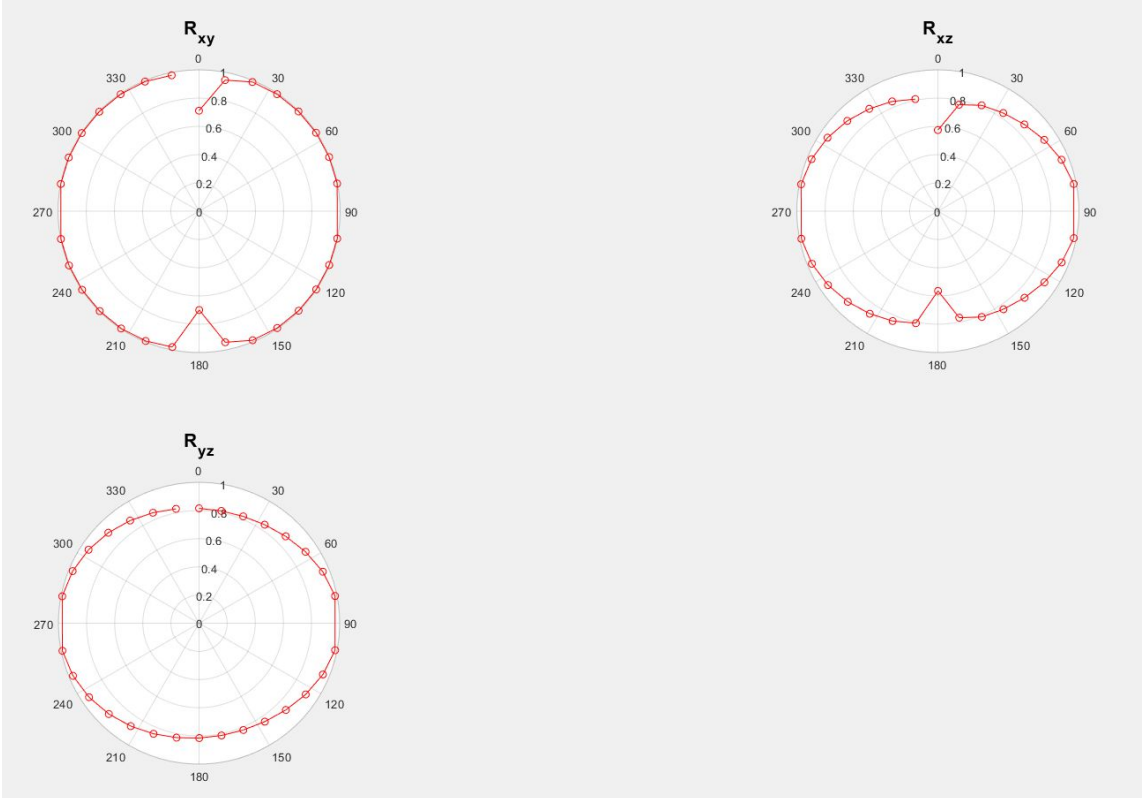


Рисунок 4.8 — Залежності взаємної кореляції помилок визначення координат цілі від кута приходу сигналу.

Висновки

За результатами моделювання, видно, що на графіках є залежність від кута приходу сигналу для СКВ помилок визначення координат цілі по осі X та Z. Для осі Y залежність відсутня, при будь якому значенні кута приходу значення СКВ для цієї осі залишаєтсья сталим.

При віддалені рухомої цілі від мережі відеокамер значення СКВ збільшується. Це можна пояснити тим, що при віддалені БПЛА від датчиків, значення кутови помилких помилок збільшуються.

При наближенні цілі до бази датчиків, СКВ для координати X стрімко зростає. Це серьозно впливає на точність вимірювань.

Тому можна додавати одну чи декілька груп відеокамер, які мають інше розташування, у необхідні сектори та потім оброблювати дані від кожної групи та об’єднувати їх у фінальний результат.

Такий підхід дозволить створити більш детальну картину руху цілі, що сприятиме ефективнішому моніторингу траєкторії рухомої цілі.

5 РОЗРОБКА АЛГОРИТМУ ФІЛЬТРАЦІЇ ПАРАМЕТРІВ РУХУ ОБ'ЄКТУ ЗА ДАНИМИ ДВОХ ВІДЕОКАМЕР У ПРЯМОКУТНІЙ СИСТЕМІ КООРДИНАТ

До завдань по визначенню параметрів різних об'єктів, що рухаються, за інформацією вимірювань траєкторії належить сімейство алгоритмів по траєкторній фільтрації. Найвідоміший та найуживаніший представником цього сімейства алгоритмів є метод калманівської фільтрації, який був поданий на розгляд у 1960 році Рудольфом Емілем Калманом. Фільтр Калмана вважається оптимальниім для процесу мінімізації середньоквадратичної помилки оцінки.

Зазвичай, під час вирішенні прикладних задач по обробці даних траєкторії, використовується декартова СК. Через це в подальшому у даній роботі буде використовуватися саме ця СК.

Перед синтезом алгоритму калманівської фільтрації спочатку опишемо математичну модель процесу, який фільтрується, та модель спостереження, що описує процес вимірювання місцеположення рухомої цілі за допомогою мережі відеокамер.

Для усіх моделей у цьому розділі використовується стохастична динамічна система, а час вважається дискретним.

$$u(k) = F(k, k-1)u(k-1) + G(k, k-1)\omega(k-1), \qquad (5.1)$$

де $u^T(k) = (x(k), \dot{x}(k), y(k), \dot{y}(k), z(k), \dot{z}(k))$ – вектор стану, який містить інфромацію про положення рухомого об'єкта у місцевій СК та його швидкість відносно кожної осі.

$$F(k, k-1) = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.2)$$

$$G(k) = \begin{bmatrix} 0.5 \cdot T^2 & 0 & 0 \\ T & 0 & 0 \\ 0 & 0.5 \cdot T^2 & 0 \\ 0 & T & 0 \\ 0 & 0 & 0.5 \cdot T^2 \\ 0 & 0 & T \end{bmatrix} \qquad (5.3)$$

Кореляційна матриця вектора гаусівських шумів збурень $\omega(k)$ має наступний вигляд:

$$Q(k) = diag\left(\sigma_{ax}^2(k), \sigma_{ay}^2(k), \sigma_{az}^2(k)\right), \quad (5.4)$$

де T — час надходження нової інформації.

Рівняння (5.1) описує динаміку руху цілі в просторі, використовуючи модель другого порядку. Ця модель враховує не лише координати цілі, але й швидкість їх зміни.

Для рівняння спостереження будемо використовувати алгоритм, який був отриманий у попередньому розділі і описує процес визначення місцеположення рухомого БПЛА за інформацією від мережі відеодатчиків.

$$u^M(k) = H(k)u(k) + \nu(k), \quad (5.5)$$

де $H(k)$ — матриця спостереження буде мати наступний вигляд:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$u^{MT}(k) = (x^T(k), y^T(k), z^T(k))$ — вектор спостереження, який містить координати цілі у декартовій СК.

Координати у цьому векторі визначаються за допомогою методу триангуляції по отриманій інформації від двох відеокамер.

$$x^M(k) = d_1^M(k)\, sin\, \varphi_1^M\,(k),$$
$$y^M(k) = d_1^M(k)\, cos\, \varphi_1^M\,(k),$$
$$z^M(k) = d_1^M(k) tg\theta_1^M(k),$$

де $d_1^M(k)$ — параметр, який можа отримати за наступною формулою:

$$d_1^M(k) = \frac{b \cdot sin\left(180^\circ - (\psi_b - \varphi_2^M(k))\right)}{sin\left((\psi_b - \varphi_2^M(k)) - (\psi_b - \varphi_1^M(k))\right)},$$

$\nu(k)$ — вектор помилок вимірювання, який містить матрицю кореляції:

$$R(k) = \begin{bmatrix} \sigma_x^2(k) & \sigma^2{}_{xy}(k) & \sigma^2{}_{xz}(k) \\ \sigma^2{}_{xy}(k) & \sigma_y^2(k) & \sigma^2{}_{yz}(k) \\ \sigma^2{}_{xz}(k) & \sigma^2{}_{yz}(k) & \sigma_z^2(k) \end{bmatrix}$$

Елементи цієї матриці можна визначити за наступними формулами:

$$\sigma_x^2(k) = \beta_1^2(k) \cdot \sigma_{\varphi_1}^2 + \beta_2^2(k) \cdot \sigma_{\varphi_2}^2$$
$$\sigma_y^2(k) = \alpha_1^2(k) \cdot \sigma_{\varphi_1}^2 + \alpha_2^2(k) \cdot \sigma_{\varphi_2}^2$$

$$\sigma_z^2(k) = \gamma_1^2(k)\sigma_{\varphi_1}^2 + \gamma_2^2(k)\sigma_{\varphi_2}^2 + \gamma_3^2(k)\sigma_{\theta_1}^2$$

$$\sigma^2{}_{xy} = \alpha_1(k) \cdot \beta_1(k) \cdot \sigma_{\varphi_1}^2 + \alpha_2(k) \cdot \beta_2(k) \cdot \sigma_{\varphi_2}^2$$

$$\sigma^2{}_{xz}(k) = \alpha_1(k) \cdot \gamma_1(k) \cdot \sigma_{\varphi_1}^2 + \alpha_2(k) \cdot \gamma_2(k) \cdot \sigma_{\varphi_2}^2$$

$$\sigma^2{}_{yz}(k) = \beta_1(k) \cdot \gamma_1(k) \cdot \sigma_{\varphi_1}^2 + \beta_2(k) \cdot \gamma_2(k) \cdot \sigma_{\varphi_2}^2$$

де $\alpha_1(k)$, $\beta_1(k)$, $\alpha_2(k)$, $\beta_2(k)$, $\gamma_1(k)$, $\gamma_2(k)$, $\gamma_3(k)$ — розраховуються по наступним формулам:

$$\alpha_1(k) = c_1(k)\cos\varphi_1(k) - d_1\sin\varphi_1(k)$$

$$\alpha_2(k) = c_2(k)\cos\varphi_1(k)$$

$$\beta_1(k) = c_1(k)\sin\varphi_1(k) + d_1(k)\cos\varphi_1(k);$$

$$\beta_2(k) = c_1(k)\sin\varphi_1(k)$$

$$\gamma_1(k) = c_1(k)tg\theta_1^M(k)$$

$$\gamma_2(k) = c_2(k)tg\theta_1^M(k)$$

$$\gamma_3(k) = \frac{d_1(k)}{\cos^2\theta_1^M(k)};$$

де $c_1(k)$, $c_2(k)$ — параметри, які розраховуються за формулами:

$$c_1(k) = -\frac{b\cdot\sin\left(180^\circ-(\psi_b-\varphi_2^M(k))\right)\cos\left(\varphi_1^M(k)-\varphi_2^M(k)\right)}{\left(\sin\left(\varphi_1^M(k)-\varphi_2^M(k)\right)\right)^2}$$

$$c_2(k) = \frac{b}{\sin\left(\varphi_1^M(k)-\varphi_2^M(k)\right)}\left(\cos\left(180^\circ-\left(\psi_b-\varphi_2^M(k)\right)\right)-\right.$$

$$\left.-\frac{\sin\left(180^\circ-(\psi_b-\varphi_2^M(k))\right)\cos\left(\varphi_1^M(k)-\varphi_2^M(k)\right)}{\sin\left(\varphi_1^M(k)-\varphi_2^M(k)\right)}\right)$$

Як зазначалося раніше, усі випадкові складові у формулі (5.1) є гаусівськими. Отже стан на початку роботи системи можна описати вектором середніх $M\big(u(0)\big) = \bar{u}(0)$ і кореляційною матрицею $P(0) = m((u(0)-\bar{u}(0))(u(0)-\bar{u}(0))^T)$.

Шуми збурень , $\omega(k)$ та шуми вимірювань $\nu(k)$ вважаються некорельованими. В результаті цього маємо наступний вираз $M\big(\omega(k)v^T(j)\big) = 0$ для усіх значень k та j.

Ціль фільтрації за допомогою фільтру Калмана полягає у тому, щоб по певній кількості вимірювань, які були зроблені послідовно,

$U^M(k) = \{u^M(1),\ u^M(2), \dots, u^M(k)\}$ знайти певне значення оцінки $\hat{u}(k/k)$ для вектору стану $u(k)$, при якому значення середнього квадрату помилки буде мінімальним. Сама помилка оцінки описується виразом $\hat{u}(k/k) = u(k) - \hat{u}(k/k)$. Оцінка $\hat{u}(k/k)$ потрапляє під критерій оптимальності при наступних умовах:

1. Оцінка незміщена $M[\hat{u}(k/k)] = M[u(k)]$
2. Слід матриці має мінімальне значення $M\left[u(k) - \hat{u}\left(\frac{k}{k}\right)^T \left(u(k) - \hat{u}\left(\frac{k}{k}\right)\right)\right] = min$

Оцінка вектора стану, який підпадає під критерії, які наведені вище, являє собою умовно середнім[10]:

$$u(k/k) = M[u(k)|U^M(k)] \tag{5.6}$$

Внаслідок цього, для розрахунку цієї оцінки треба вирахувати апостеріорну щільність розподілу $f[u(k)|U^M(k)]$, яка здобувається, використовуючи формулу Баєса:

$$f[u(k)|U^M(k)] =$$

$$f[u(k)|U^M(k), u^M(k)\ ] = \frac{f[u(k)|U^M(k-1)\ ] \cdot f[u^M(k)|u(k), u^M(k-1)\ ]}{f[u^M(k)|U^M(k-1)]} \tag{5.7}$$

Вираз $f[u^M(k)|u(k), u^M(k-1)\ ]$ дорівнює $f[u^M(k)|u(k)]$, бо, як зазначалося раніше, $v(k)$ — білий шум Гауса. Виходячи з цього, коли значення $u(k)$ фіксоване, то вектор $u^M(k)$ не залежить від попередніх вимірювань $U^M(k-1)$. Тому з виразу спостереження маємо наступне твердження:

$$f[u^M(k)|u(k)] = N\{H(k)x(k), R(k)\}, \tag{5.8}$$

де $R(k)$ — кореляційна матриця помилки під час вимірювання,

$N\{X, Y\}$ — формат запису гаусівської щільності ймовірності з вектором середнього X та кореляційною матрицею Y.

Перед розрахунком лівого члена чисельника формули (5.7) введемо наступні позначення:

$$M[u(k)|U^M(k-1)] = \hat{u}(k|k-1), \tag{5.9}$$

$$M\{[u(k) - \hat{u}(k|k-1)][u(k) - \hat{u}(k|k-1)|U^M(k-1)]\} =$$

$$= M\{\tilde{u}(k|k-1)\tilde{u}^T(k|k-1)|U^M(k-1)\} = P(k|k-1) \tag{5.10}$$

де $P(k|k-1)$ — кореляційна матриця помилок екстраполяції,

$\hat{u}(k|k-1)$ — оцінка прогнозу на наступному кроці.

Підставивши формули (5.9) і (5.10) у рівняння (5.1) отримаємо:

$$\hat{u}(k|k-1) = M\{F(k,k-1)u(k-1) + G(k,k-1) \cdot$$
$$\cdot\ \omega(k-1)|U^M(k-1)\} = F(k,k-1)M\{u(k-1)|U^M(k-1)\} = \quad (5.11)$$
$$= F(k,k-1)\hat{u}(k-1|k-1) \cdot$$

$$P(k|k-1) = M\{\tilde{u}(k|k-1)\tilde{u}^T(k|k-1)|U^M(k-1)\} =$$
$$= F(k,k-1)P(k-1|k-1)F^T(k,k-1) + \quad (5.12)$$
$$+G(k,k-1)Q(k-1)G^T(k,k-1)$$

де $P(k-1|k-1)$ — кореляційна матриця помилок під час фільтрації на попередньому кроці.

Вважаємо, що середнє значення вектору шумів збудження нульове. Крім того, цей вектор та вектор стану некорельовані між собою. Тоді,

$$f[u(k)|U^M(k-1)\,] = N\{\hat{u}(k|k-1), F(k,k-1)P(k-1,k-1) \cdot$$
$$F^T(k,k-1) + G(k,k-1)Q(k-1)G^T(k,k-1)\} \quad (5.13)$$

Перед розрахунком параметрів $f[u^M(k)|U^M(k-1)]$ вважатимемо, що вектор стану та шум спостереження некорельовані між собою, та, використовуючи, рівняння спостереження, отримаємо:

$$M\{u^M(k)|U^M(k-1)\} = M\{H(k)u(k) + v(k)|U^M(k-1)\} =$$
$$= H(k)\hat{u}(k|k-1) \quad (5.14)$$

$$M\{[u^M - H(k)\hat{u}(k|k-1)][u^M(k) - H(k)\hat{u}(k|k-1)]^T =$$
$$M\{L(k|k-1)L^T(k,k-1)\} = P_L(k) = H(k)P(k,k-1) \cdot \quad (5.15)$$
$$\cdot H^T(k) + R(k)$$

де $L(k|k-1) = u^M(k) - H(k)\,\hat{u}(k|k-1)$ — оновлюючий процес.

Отже, отримуємо наступне:

$$f[u^M(k)|U^M(k-1)] = N\{H(k)\hat{u}(k|k-1), H(k)P(k|k-1) \cdot$$
$$\cdot H^T(k) + R(k))\} \quad (5.16)$$

Підсумовуючи вищевикладене, можна констатувати, що при підстановці формул (5.16), (5.13) та (5.8) у формулу (5.7) можна показати, що апостеріорна щільність ймовірності являє собою гаусівським вектором $\hat{u}(k|k)$. Це також являє собою оцінкою вектора стану, який розраховується рекурсивно.

Отже, для реалізації алгоритму, який вважається оптимальним, оцінювання вектору стану треба використати наступну систему рекурсивних рівнянь:

$$P(k/k-1) = F(k,k-1)P(k-1/k-1)\cdot \\ \cdot F^T(k,k-1) + G(k,k-1)Q(k-1)G^T(k,k-1) \tag{5.17}$$

$$K(k) = P(k/k)H^T(k)R^{-1}(k) = \\ = P(k/k-1)H^T(k)(H(k)P(k/k-1)H^T(k) + R(k))^{-1} \tag{5.18}$$

$$P(k/k) = P(k/k-1) - P(k/k-1)H^T(k)\cdot \\ \cdot [H(k)P(k/k-1)H^T + R(k)]^{-1}H(k)P(k/k-1) \tag{5.19}$$

$$\hat{u}(k/k) - F(k,k-1)\hat{u}(k-1/k-1) + K(k)\cdot \\ \cdot [u^M(k) - H(k)F(k,k-1)\hat{u}(k-1/k-1)] = \\ = \hat{u}(k/k-1) + K(k)L(k,k-1) \tag{5.20}$$

де $P(k/k-1)$ — кореляційна матриця помилки прогнозу; $P(k/k)$ — кореляційна матриця помилки оцінки; $K(k)$ — коефіцієнт підсилення фільтру Калмана, рівняння (5.20) — рівняя оцінки.

Система рівнянь (5.17)-(5.20) являє собою фільтр Калмана. Для цієї системи потрібні наступні початкові умови:

$$\hat{u}(0/0) = \bar{u}(0) \tag{5.21}$$

$$P(0/0) = P(0) \tag{5.22}$$

Його структурна схема зображена на рис. 5.1.

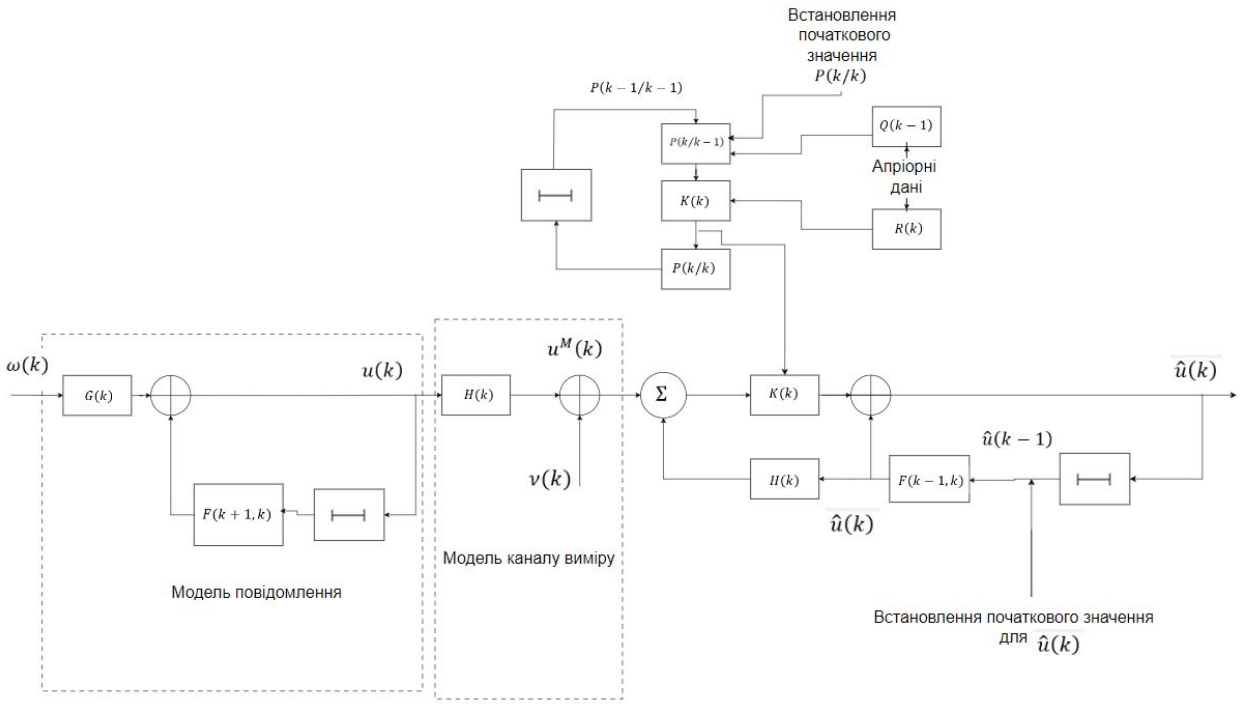


Рисунок 5.1 — Структурна схема фільтру Калмана

Ціль пошуку оптимального прийнятого повідомлення можна вирішити за допомогою використання фільтра Калмана. Спочатку йде ініціалізація початкових значень для $\hat{u}(0)$ та $P(0)$. Вектор $u(k)$, що містить стан системи, містить в собі повідомлення, яке було передане.

Під час моделювання цей вектор представляється у вигляді випадкового процесу, який був отриманий після пропущення білого гаусівського шума $\omega(k)$ через лінійний фільтр. У загальному випадку присутні ще компоненти, які залежать від часу та шумів збурення. Перший компонент визначається перехідною матрицею системи $F(k+1,k)$, а другий — матрицею шумів збурення $G(k+1,k)$. Потім на виміряний сигнал додається шум вимірювання.

Оцінки вектору, що містить стан системи, розраховується рекурсивно, продовж надходження все нової інформації. Крім того, як видно по рівнянню оцінки (5.20) для вираховування теперішнього значення оцінки, треба тримати у пам’яті лише розраховане значення оцінки на попередньому кроці.

У випадку використання дискретного фільтру Калмана також треба задати початкові умови. Після перших двох вимірювань вектор початкової оцінки у даній роботі має наступний вигляд:

$$\hat{u}^T(k) = (x^M(1), \frac{x^M(1)-x^M(0)}{T}, y^M(1), \frac{y^M(1)-y^M(0)}{T},$$
$$, z^M(1), \frac{z^M(1)-z^M(0)}{T}) \quad (5.23)$$

Для початкової оцінки її кореляційна матриця має наступний вигляд

$$\begin{bmatrix}
\sigma_x^2(1) & \frac{\sigma_x^2(1)}{T} & \sigma_{xy}^2(1) & \frac{\sigma_{xy}^2(1)}{T} & \sigma_{xz}^2(1) & \frac{\sigma_{xz}^2(1)}{T} \\
\frac{\sigma_x^2(1)}{T} & \frac{\sigma_x^2(1)+\sigma_x^2(0)}{T^2} + \sigma_{ax}^2 T^2 & \frac{\sigma_{xy}^2(1)}{T} & \frac{\sigma_{xy}^2(1)+\sigma_{xy}^2(0)}{T^2} & \frac{\sigma_{xz}^2(1)}{T} & \frac{\sigma_{xz}^2(1)+\sigma_{xz}^2(0)}{T^2} \\
\sigma_{xy}^2(1) & \frac{\sigma_{xy}^2(1)}{T} & \sigma_y^2(1) & \frac{\sigma_y^2(1)}{T} & \sigma_{yz}^2(1) & \frac{\sigma_{yz}^2(1)}{T} \\
\frac{\sigma_{xy}^2(1)}{T} & \frac{\sigma_{xy}^2(1)+\sigma_{xy}^2(0)}{T^2} & \frac{\sigma_y^2(1)}{T} & \frac{\sigma_y^2(1)+\sigma_y^2(0)}{T^2} + \sigma_{ay}^2 T^2 & \frac{\sigma_{yz}^2(1)}{T} & \frac{\sigma_{yz}^2(1)+\sigma_{yz}^2(0)}{T^2} \\
\sigma_{xz}^2(1) & \frac{\sigma_{xz}^2(1)}{T} & \sigma_{yz}^2(1) & \frac{\sigma_{yz}^2(1)}{T} & \sigma_z^2(1) & \frac{\sigma_z^2(1)}{T} \\
\frac{\sigma_{xz}^2(1)}{T} & \frac{\sigma_{xz}^2(1)+\sigma_{xz}^2(0)}{T^2} & \frac{\sigma_{yz}^2(1)}{T} & \frac{\sigma_{yz}^2(1)+\sigma_{yz}^2(0)}{T^2} & \frac{\sigma_z^2(1)}{T} & \frac{\sigma_z^2(1)+\sigma_z^2(0)}{T^2} + \sigma_{az}^2 T^2
\end{bmatrix}$$

Висновки

Розробка алгоритму фільтрації параметрів руху об’єкту за даними двох відеокамер у прямокутній СК на основі фільтра Калмана має наступні переваги:

- алгоритм має рекурентний характер**,** що робить його зручним для реалізації на ЕОМ;
- оцінка стану $\hat{u}(k)$ є лінійною відносно спостережень;
- оскільки модель вимірювання положення координат об’єкту в прямокутній СК є нестаціонарною, алгоритм забезпечує мінімум СКВ помилки фільтрації для нестаціонарних процесів;
- алгоритми фільтрації легко розповсюджується на багатоканальні системи, що є важливим при отриманні даних від мережі відеокамер.

6 АНАЛІЗ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО АЛГОРИТМУ ТРАЕКТОРНОЙ ФІЛЬТРАЦІЇ ШЛЯХОМ СТАТИСТИЧНОГО МОДЕЛЮВАННЯ

Моделювання процесу траекторної фільтрації відбувалось у прямокутній СК, як зазначалося у попередньому розділі. Симуляція проводилась на ЕОМ у програмному забезпеченні Matlab.

Кореляційна матриця вектора помилок вимірювання $v(k)$ має наступний вигляд:

$$R(k) = \begin{bmatrix} \sigma_x^2(k) & 0 & 0 \\ 0 & \sigma_y^2(k) & 0 \\ 0 & 0 & \sigma_z^2(k) \end{bmatrix}$$

Моделювання у середовищі Matlab відбувалося при наступних параметрах:

- Помилка вимірювання азимут та кута місця для обох відеокамер становили 0.1 градуса.
- Час надходження інформації – 0.1 с.
- Прискорення по осі X та Y становить $\sigma_{ax} = \sigma_{ay} = 2.5$ м/с^2.
- Прискорення по осі Z становить $\sigma_{az} = 0.5$ м/с^2.

Кількість модельних траєкторій при застосуванні методу Монте-Карло дорівнювало 100. Кількість тактів траєкторії дорівнює 52. Два перших такта використовуються для формування початкових умов для алгоритма калманівської фільтрації.

На рис. 6.1 показана траєкторія руху БПЛА у площині XY, а на рис. 6.2 зображений графік зміни висоти БПЛА.

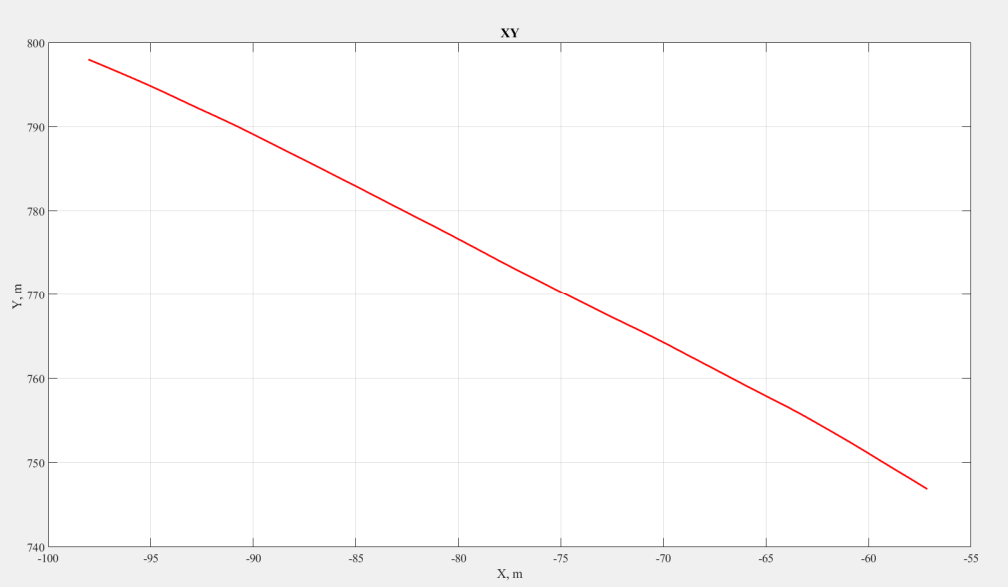


Рисунок 6.1 — Траєкторія руху БПЛА у площині XY

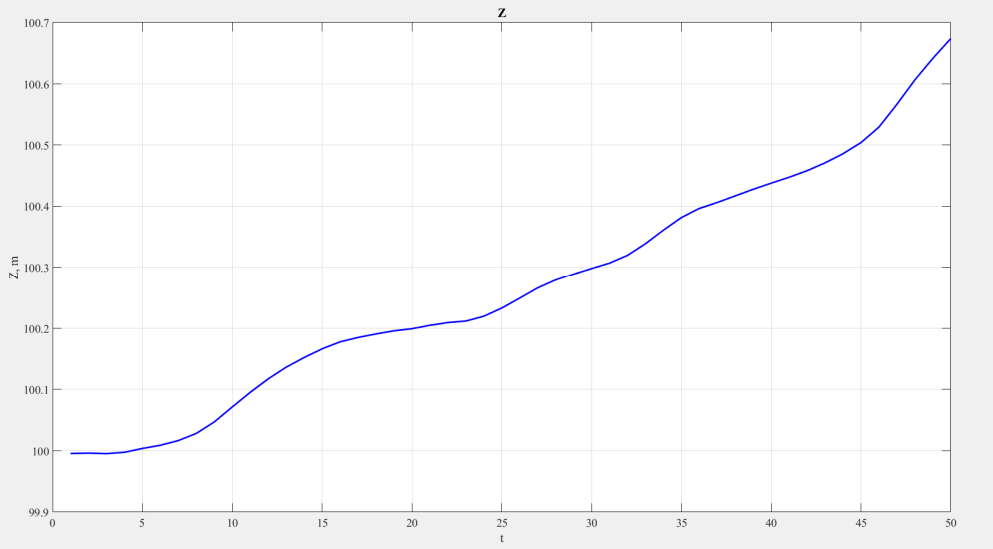


Рисунок 6.2 — Графік зміни висоти БПЛА

На рис. 6.3 наведено наступні характеристики помилки вимірювання координати X: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

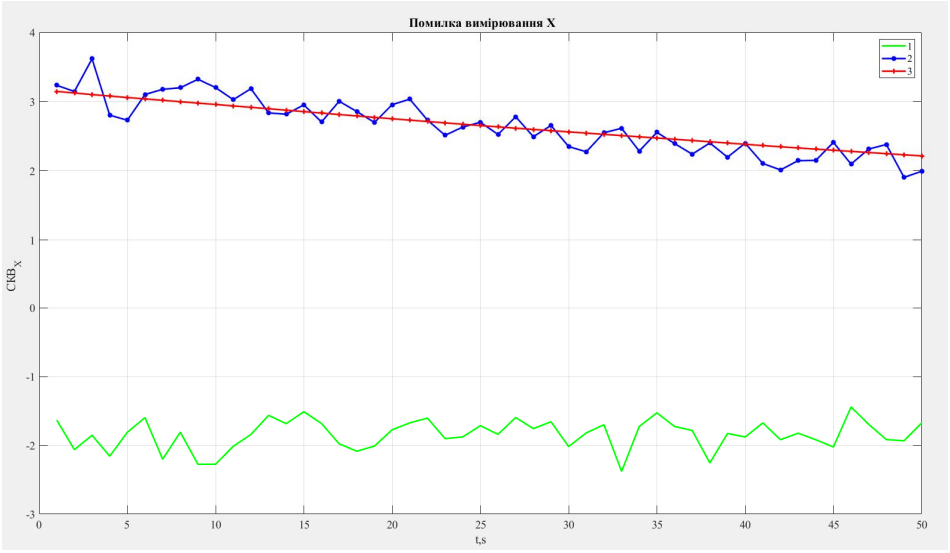


Рисунок 6.3 — Математичне очікування та СКВ помилок вимірювання координат X

На рис. 6.4 наведено наступні характеристики помилки вимірювання координати Y: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

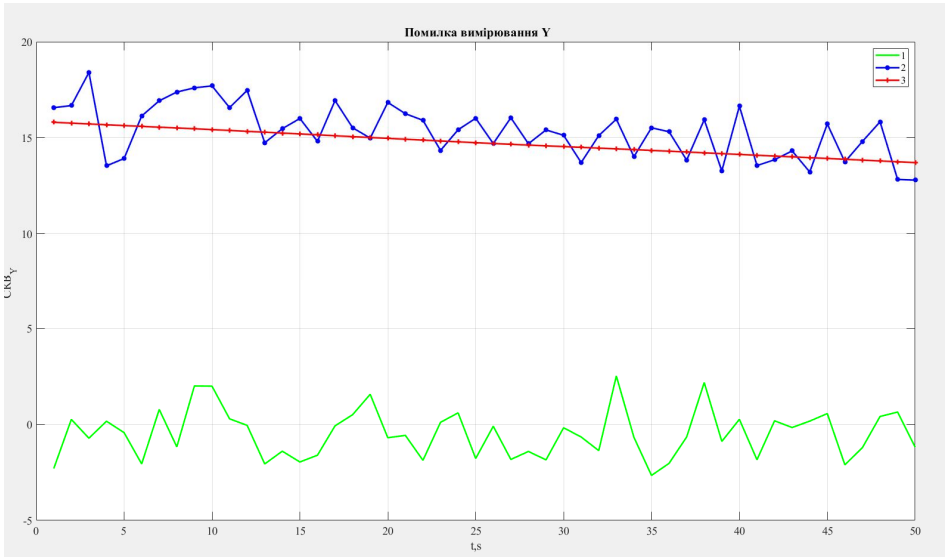


Рисунок 6.4 — Математичне очікування та СКВ помилок вимірювання координати Y

На рис. 6.5 наведено наступні характеристики помилки вимірювання координати Z: математичне очікування (зелена крива), СКВ (синя крива), які

отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

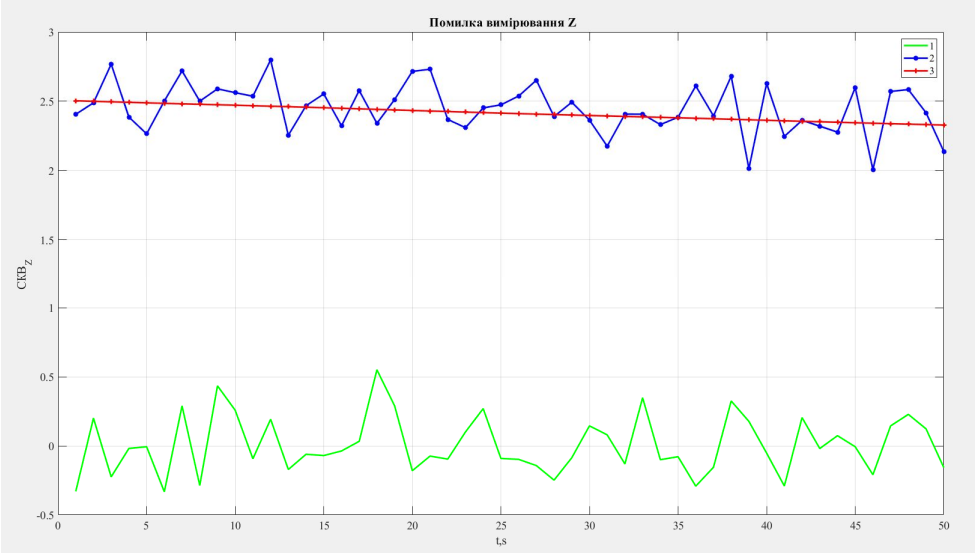


Рисунок 6.5 — Математичне очікування та СКВ помилок вимірювання координати Z

На рис. 6.6 наведено наступні характеристики помилки оцінки координати X: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

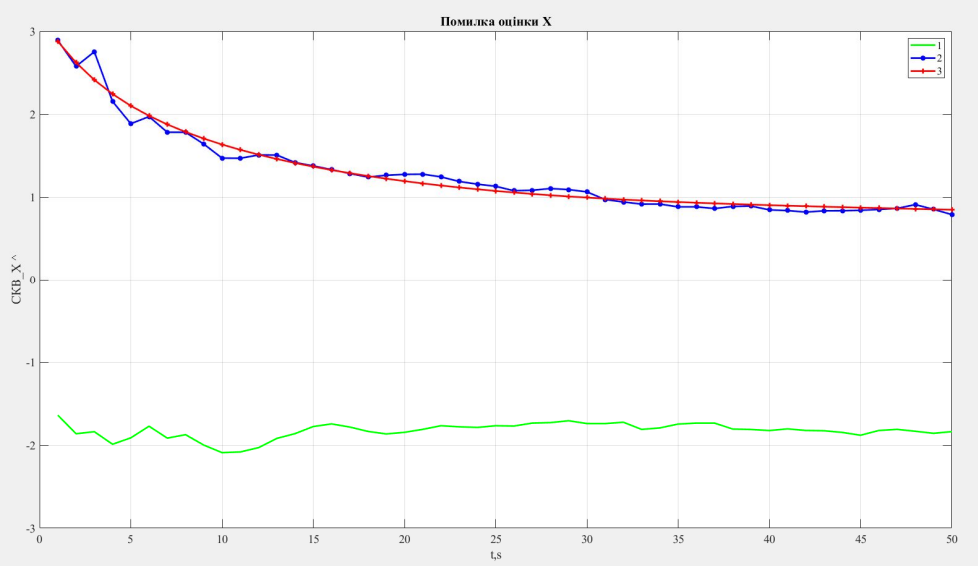


Рисунок 6.6 —Математичне очікування та СКВ помилок оцінки координати X

На рис. 6.7 наведено наступні характеристики помилки оцінки координати Y: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

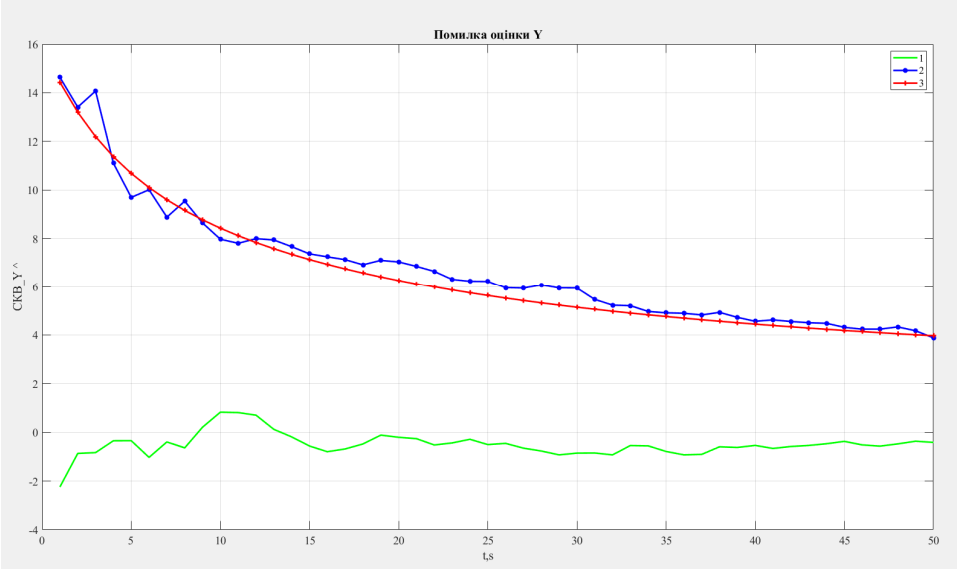


Рисунок 6.7 — Математичне очікування та СКВ помилок оцінки координати Y

На рис. 6.8 наведено наступні характеристики помилки оцінки координати Z: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

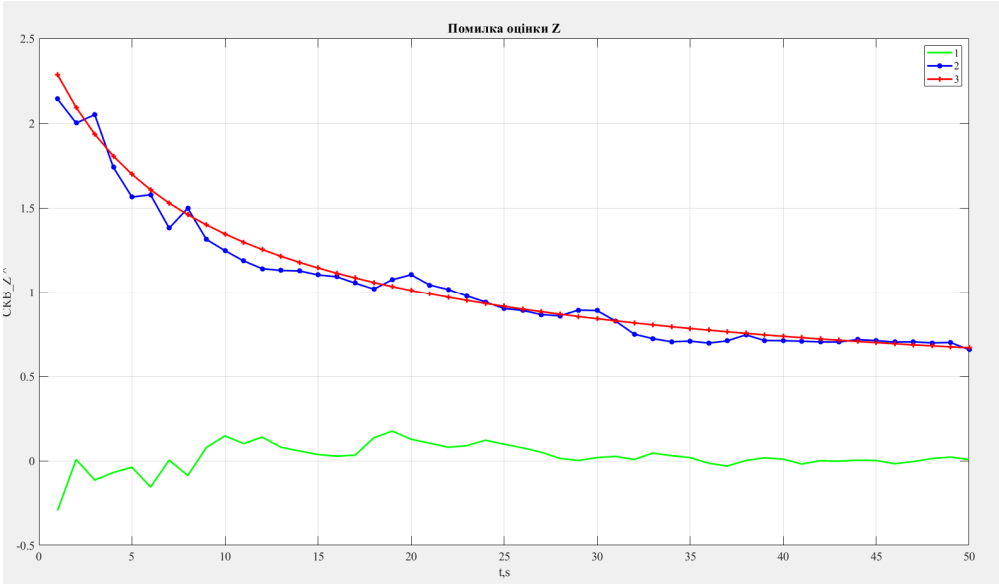


Рисунок 6.8 — Математичне очікування та СКВ помилок оцінки координати Z

На рис. 6.9 наведено наступні характеристики помилки прогнозу координати X: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

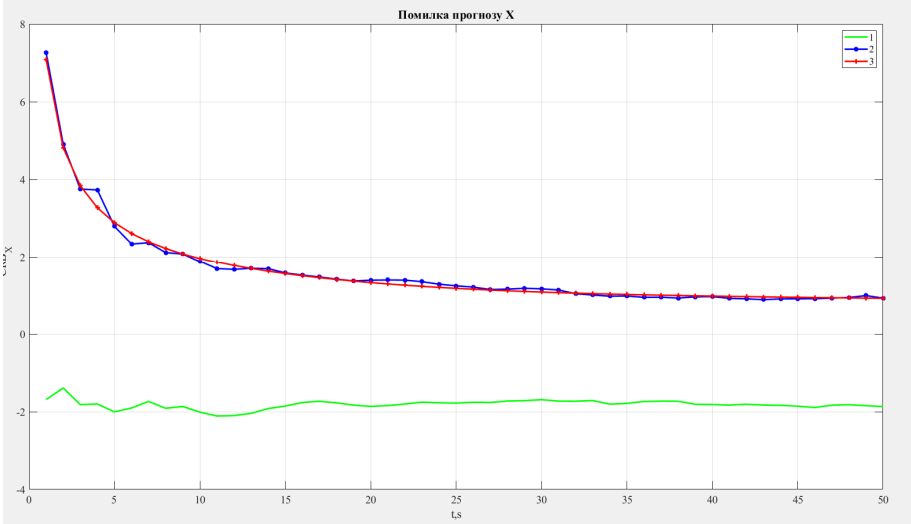


Рисунок 6.9 — Математичне очікування та СКВ помилок прогнозу координати X

На рис. 6.10 наведено наступні характеристики помилки прогнозу координати Y: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

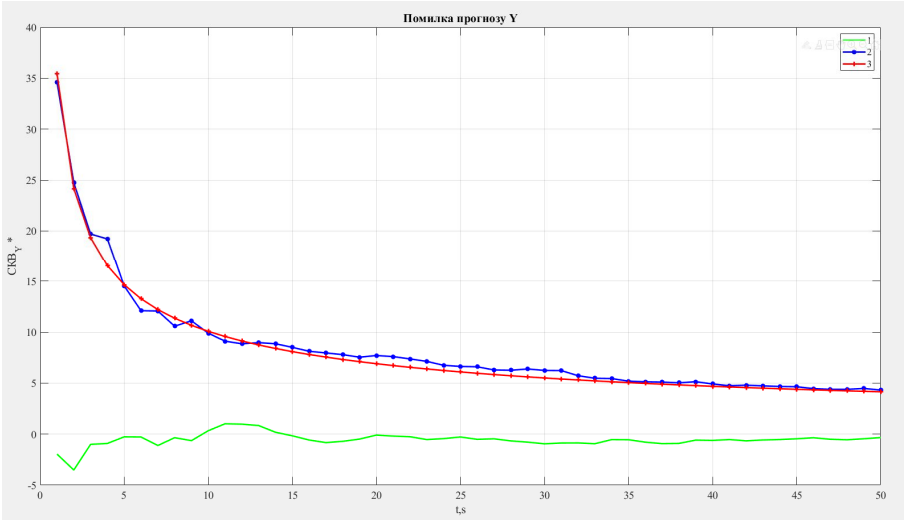


Рисунок 6.10 — Математичне очікування та СКВ помилок прогнозу координати Y

На рис. 6.11 наведено наступні характеристики помилки прогнозу координати Z: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

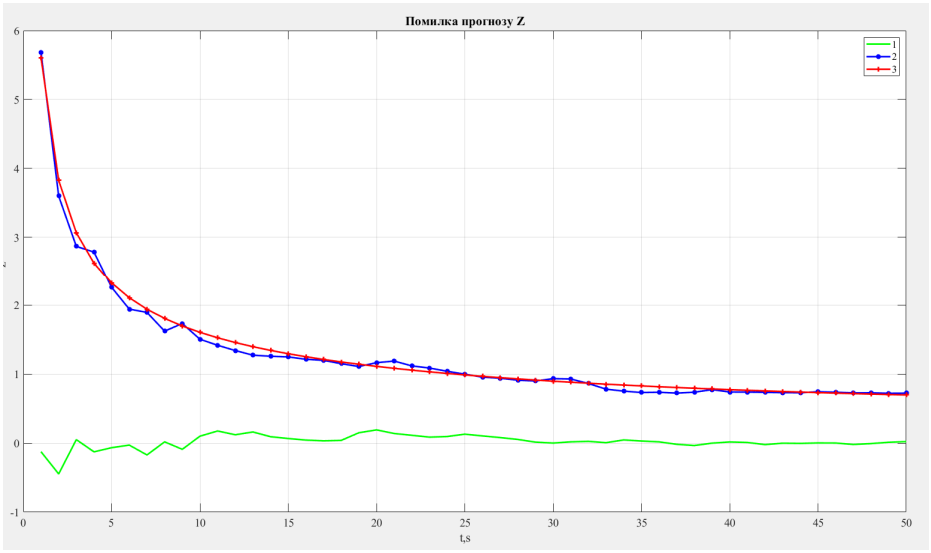


Рисунок 6.11 — Математичне очікування та СКВ помилок прогнозу координати Z

На рис. 6.12 наведено наступні характеристики помилки оцінки швидкості по осі координати X: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

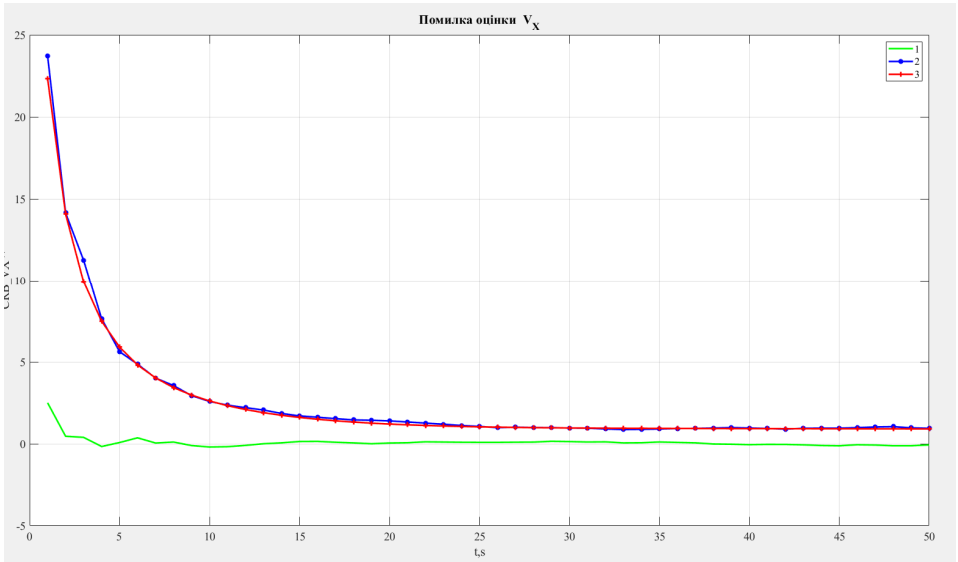


Рисунок 6.12 — Математичне очікування та СКВ помилок оцінки швидкості за координатою X

На рис. 6.13 наведено наступні характеристики помилки оцінки швидкості по осі координати Y: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

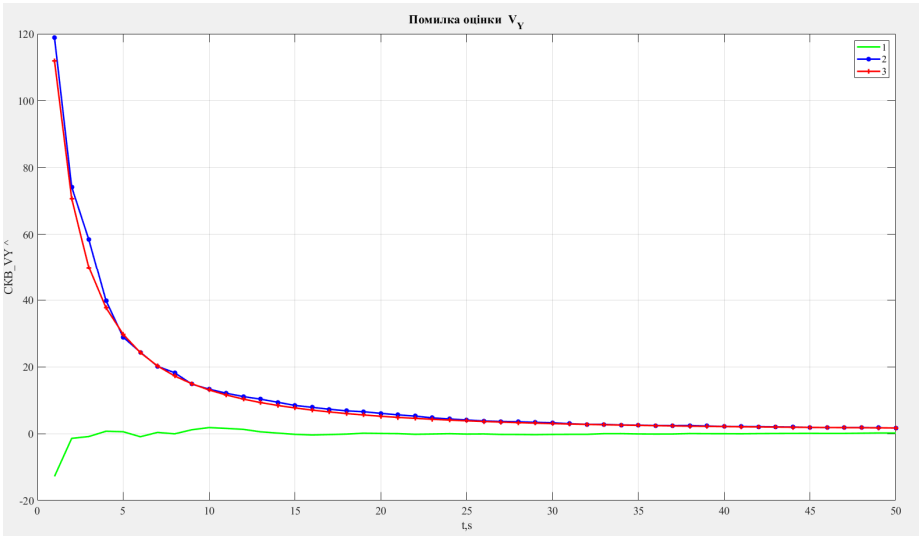


Рисунок 6.13 — Математичне очікування та СКВ помилок оцінки швидкості за координатою Y

На рис. 6.1 наведено наступні характеристики помилки оцінки швидкості по осі координати Z: математичне очікування (зелена крива), СКВ (синя крива), які отримані за допомогою методу Монте-Карло, а СКВ (червона крива), що розраховано калманівським фільтром.

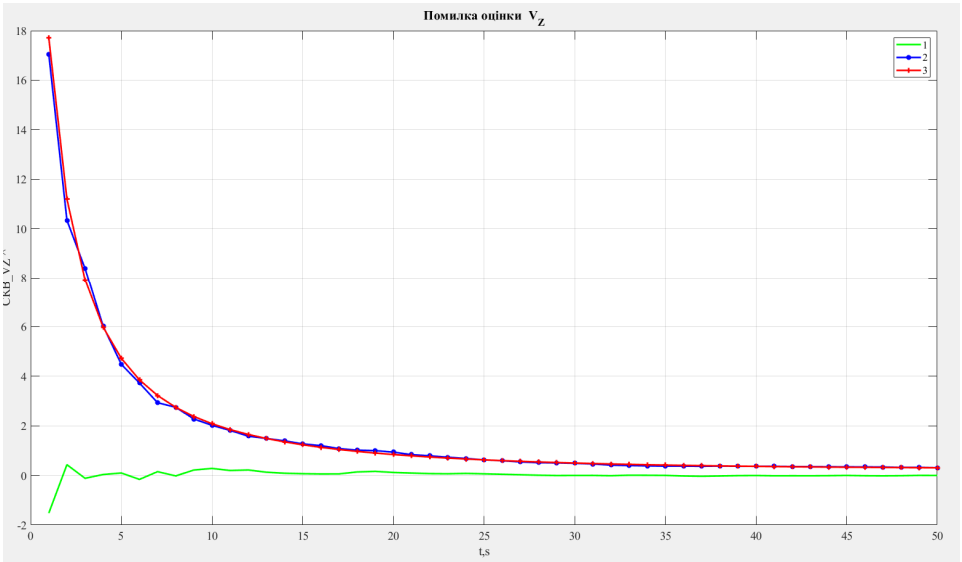


Рисунок 6.14 — Математичне очікування та СКВ помилок оцінки швидкості за координатою Z

Висновки

Після моделювання можна прийти до наступних висновків:

- На усіх графіках значення СКВ, які були отриманні фільтром Калмана, та фактичні значення СКВ асимптотично наближаються один до одного, що свідчить про правильну роботу алгоритму.
- На графіках видно, що значення математичного очікування не одразу починає приближатися до нуля. Це можна пояснити перехідним процесом роботи фільтра.
- Розроблений фільтр дозволяє зменшити СКВ похибок прогнозу і оцінки координат цілі відповідно в 6.9 рази для осі X, в 8 рази для осі Y, в 7.5 рази для осі Z і 3.2 рази для осі X, в 3.3 рази для осі Y, в 3.2 рази для осі Z рази у порівнянні у порівнянні з СКВ похибок вимірювання. СКВ похибок оцінки швидкості цілі також зменшується в 23.3 рази для осі X, в 60.5 рази для осі Y, в 55.4 рази для осі Z. При віддалені рухомої цілі від мережі відеокамер значення СКВ оцінювання параметрів руху цілі збільшується.

7 АНАЛІЗ МОЖЛИВОСТІ ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ РОЗРОБЛЕНОГО АЛГОРИТМУ ТРАЄКТОРНОЇ ОБРОБКИ ІНФОРМАЦІЇ ЗА ДАНИМИ МЕРЕЖІ ВІДЕОДАТЧИКІВ

Оскільки, крім самих відеокамер, виконуються ще математичні розрахунки для імплементації фільтру Калмана на ЕОМ. Ці дії виконуються в режимі реального часу під час виявлення рухомого об’єкта. Тому треба якісно підібрати мінімально необхідну кількість пам’яті для коректної роботи усієї системи.

Необхідний обсяг пам'яті визначається розмірністю відповідних векторів і матриць, що входять в алгоритм калманівської фільтрації. Занесемо у таблицю 6.1 відповідні вектори і матриці, що необхідну кількість пам’яті з довільним доступом.

Таблиця 6.1 Інформація про кількість пам’яті для векторів та матриць

Зміна	Розмірність	Мінімальна кількість комірок пам’яті
$u^{\mathrm{M}}(k)$	$s \times 1$	s
$\hat{u}(k / k-1)$	$n \times 1$	n
$\hat{u}(k / k)$	$n \times 1$	n
$K(k)$	$n \times s$	ns
$P(k / k-1)$	$n \times n$	n^2
$P(k / k)$	$n \times n$	n^2
$H(k)$	$n \times s$	ns
$R(k)$	$s \times s$	s^2
$G(k)$	n$\times r$	nr
$Q(k)$	$r \times r$	r^2
$F(k, k-1)$	$n \times n$	n^2

Однією з основних характеристик алгоритму є час обчислень, затрачуваний на один крок калманівської фільтрації. Цей час визначається як швидкодією ЕОМ, так і загальним числом операцій, необхідних при обчисленні. Більш точна його

оцінка вимагає знання системи команд конкретної ЕОМ, на якій реалізується даний фільтр.

Під час роботи алгоритму обіг матриці за стандартним алгоритмом Гауса здійснюється за s^3 операцій множення та додавання, де s – розмірність матриці. В результаті отримуємо наступні формули для кількості операцій множення (6.1) та кількості операцій додавання (6.2):

$$M(n,s,r) = 3n^3 + 2sn^2 + 2ns^2 + n^2r + nr^2 + 2ns + s^3 + n^2 \qquad (6.1)$$

$$A(n,s,r) = 3n^3 + 2n^2s + 2ns^2 + n^2r + nr^2 + s^3 - 2n^2 - nr \qquad (6.2)$$

M(6, 3, 3) = 1233

A(6, 3, 3) = 1071

Тоді в сумі отримаємо сумарну кількість операцій, яка становить 69120. При умові, що за секунду надсилаються камерами 30 кадрів, то максимальна орієнтовна кількість операцій, що обробляється за секунду становить 69120. Мінімальна кількість комірок пам’яті становить 183.

Для функціонування усієї системи виявлення та відстеження рухомих об’єктів можна обрати мікроконтролер STM32F205ZF, частота якого може досягати 120 МГц. Структурна схема мікроконтролеру зображена на рис. 7.1

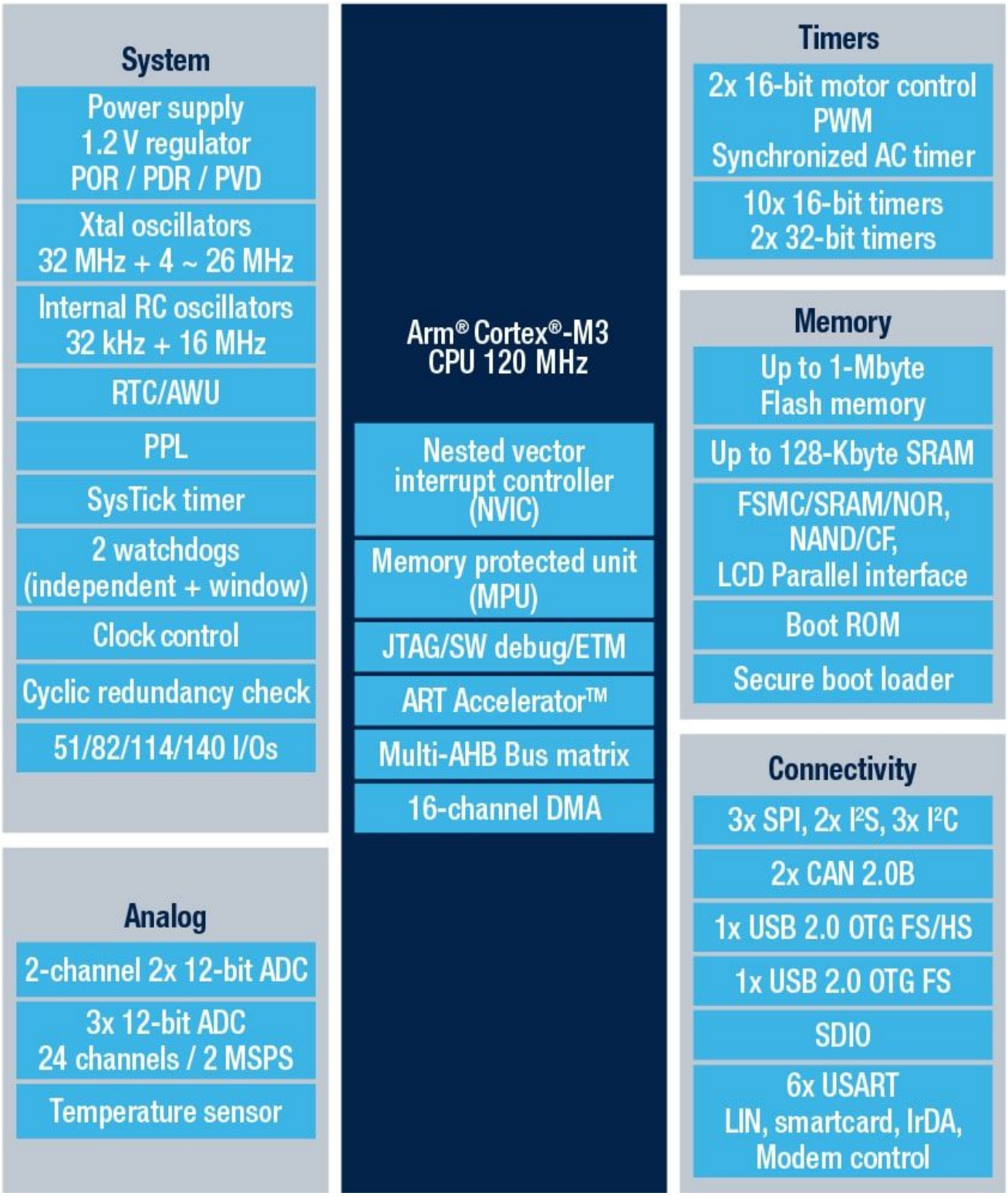


Рисунок 7.1 — Структурна схема STM32F205ZF

У випадку збільшення кількості камер є два можливі варіанти:

- Вертикальне масштабування — підбор мікроконтролера з більшими технічними характеристиками

- Горизонтальне масштабування — створення нових груп відеодатчиків у мережі, які складаються с однакових камер і однакових мікроконтролів, які оброблють дані у рамках однієї групи з двох датчиків.

Алгоритми калманівської фільтрації легко розповсюджується на багатоканальних систем, що дозволяє додавати відеокамери з іншими базами та потім об’єднувати отримані вимірювання у єдиний фінальний результат. Такий підхід дозволить створити більш детальну картину руху цілі, що сприятиме ефективнішому моніторингу траєкторії рухомої цілі.

Висновки

Таким чином, для реалізації процесу імплементації фільтру Калмана для двох відеокамер потрібно використовувати мікроконтролер STM32F205ZF від компанії STMicroelectronics.

У випадку збільшення кількості відеокамер необхідно підібрати інший мікроконтролер з більшими технічними характеристиками, цей процес називається вертикальне масштабування, або збільшувати кількість груп, які складаються, з двох відеокамер. Даний випадок також відомий як горизонтальне масштабування.

8 РОЗРОБКА СТАРТАП ПРОЕКТУ

Головна мета стартап проекту є підвищення точності виявлення рухомої цілі та покращення якості виявлення БПЛА, що рухається і фіксується кожною відеокамерою, які знаходяться у мережі.

У цьому розділі буде зроблений маркетинговий аналіз, який є найпершими етапом повноцінного процесу розробки стартапу.

8.1 Ідея проекту та її опис

Ім’я автора: Кот Максим

Назва проєкту: Алгоритми траєкторної обробки інформації за даними мережі відеодатчиків

Анотація: Під час виконання дисертації був розроблений алгоритму виявлення та відстеження рухомого БПЛА за даними мережі відеодатчиків.

Ідея стартапу полягає у підвищенні ефективності роботи алгоритму та здійснення пристроїв, які працюють на основі розробленого алгоритму.

Термін для повноцінної реалізації проєкту: для реалізації усіх поставлених задач потрібно 16 місяців.

Ресурсі, які необхідні для реалізації:

- Грошові ресурси: необхідне технічне обладнання (40000 грн), купівля ліцензійного програмного забезпечення (50000 грн)
- Інтелектуальні ресурси: праця конструктора (20000 грн), праця проектного менеджера (30000 грн), праця розробника програмного забезпечення (40000 грн)

Проблема, яку вирішує даний проект:

Задачі стартап проєкту: До основних завдань можна віднести наступні:

- **Мінімізація** - зменшення витрат розрахункових ресурсів на виявлення рухомого об'єкту.

- **Оптимізація часу виявлення** - забезпечення максимально швидкого виявлення рухомого об'єкту для підвищення загальної швидкодії системи.
- **Покращення якості виявлення цілі** - підвищення точності і надійності виявлення цілі, яка рухається, в різних умовах освітлення та за різних швидкостей руху. Це включає адаптацію системи до змін у зовнішніх умовах, що дозволяє зменшити ймовірність помилкових спрацювань та підвищити точність ідентифікації об'єктів.
- **Збільшення сектору виявлення та відслідковування** - розширення зони, в якій можливо виявляти та відслідковувати рухомі БПЛА

Очікуваний результат: У фіналі виконання даного стартап проекту очікується підвищення швидкодії, якість виявлення рухомих цілей відеокамерами, які об’єднанні у мережу, та збільшення сектору спостереження.

У таблиці 8.1.1 наведений опис стартап проєкту.

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Збільшення продуктивності алгоритму та його якості	Цивільна сфера	Ціна, швидке виявлення та відстеження траєкторії рухомого об’єкта
	Військова сфера	
	Приватний бізнес	

8.2 Технологічний аудит ідеї проекту

У рамках цього підрозділу буде проведений аудит доступних технологій. В результаті стане відомо чи можлива фактична реалізація задач, які ставляться для цього стартам проекту.

Таблиця 8.2.1 - Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології та реалізації	Наявність технологій	Доступність технологій
1	Траєкторна обробка інформації за даними мережі відеодатчиків	Мікроконтролери, відеокамери.	Доступні	Доступні

Таким чином, необхідні технологія для реалізації стартап проекту доступні.

8.3 Аналіз ринкових можливостей

У рамках даного підрозділу визначимо перелік можливостей. Їх потім у майбутньому буде використано для реалізації усіх цілей проекту.

У таблиці 8.3.1 надано характеристику потенційного ринку.

№ п/п	Показники стану (найменування)	Характеристика
1	Динаміка ринку (якісна оцінка)	розвивається
2	Швидке виявлення рухомих об’єктів	Мережа відеодатчиків буде виявляти рухомі об’єкти
3	Висока продуктивність	Швидкість фінального пристрою висока
4	Середня норма рентабельності в галузі (або по ринку), %	35
5	Кількість головних гравців	3

Отже, даний стартап проект є перспективним та прибутковим.

Зобразимо на таблиці 8.3.2 інформацію з потенційними групами населення серед цільової аудиторії.

Таблиця 8.3.2 – Інформація про ймовірних клієнтів

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відміності у Поведінці різних потенційних цільових груп клієнта	Вимоги споживачів до товару
1	Швидке та якісне виявлення та відстеження рухомого об’єкта	Військові, приватний бізнес та силові структури	Потребують оптимальну ціну та якість для товару	1. Легкість налаштування 2. Швидка робота 3. Наявність техпідтримки 4. Наявність гарантії

В результаті аналізу встановлено, що фінальний продукт стартап проекту задовольнить усі необхідні вимоги цільової аудиторії.

8.4 Розробка ринкової програми стратегії проекту

У цьому розділі зробимо порівняння ймовірних клієнтів серед цільової аудиторії. Опис кожної групи знаходиться у таблиці 6.4.

Таблиця 8.4.1 – Опис груп потенційних клієнтів

№ п/п	Опис групи потенційних клієнтів	Готовність споживачів користуватися продуктом	Орієнтований попит в межах цільової групи	Інтенсивність конкуренції	Простота вхожу у сегмент
1	Військові	так	високий	висока	висока
2	Сильові структури	так	високий	висока	висока

Продовженння таблиці 8.4.1

№ п/п	Опис групи потенційних клієнтів	Готовність споживачів користуватися продуктом	Орієнтований попит в межах цільової групи	Інтенсивність конкуренції	Простота вхожу у сегмент
3	Компанії та корпорації	так	середній	середня	середня
4	Цивільні	так	низький	середня	середня

Військові та силові структури будуть найбільше зацікавлиними у фінальному продукту даного стартап проекту. Дані групи мають постійний високий попит для виявлення та відстеження рухомих цілей.

8.5 Висновки

У результаті виконання аналізу було встановлено, що даний проект є вигідним та перспективним. Серед потенційних клієнтів обрано силові структури та військових, бо у них постійно високий попит на пристрої, які пропонують швидке та якісне виявлення та відстеження рухомих цілей за даними мережі відеокамер.

ВИСНОВКИ

У ході виконання даної роботи було розроблено алгоритми виявлення і оцінювання параметрів руху БПЛА за даними мережі відеодатчиків, яка складається із двох камер.

Виявлення та відслідковування БПЛА за допомогою даних від мережі відеодатчиків в режимі реального часу є актуальною проблемою у теперішньому часу у військовій і та інших сферах людської діяльності.

Оптичне виявлення БПЛА знаходить все більше застосування, особливо за останні роки. Це пояснюється зростанням популярності камер, що використовуються для виявлення, завдяки їхній малій вазі, простоті експлуатації та низькому енергоспоживанню. Крім того, камери здатні збирати величезну кількість даних, що робить їх незамінними для забезпечення потрібної інформації, в тому числі і ідентифікації цілей, для подальшого аналізу та вжиття заходів.

Одним із способів оптичного виявлення об'єктів в режимі реального часу є метод міжкадрових різниць. Цей метод має низку переваг, серед яких головною є його простота у впровадженні. Окремо слід виділити одну з його реалізацій, яка базується на використанні різниці трьох кадрів. При використанні цієї реалізації відсутні дублікати виявлених об’єктів при різних співвідношеннях кольору цілі та фону. При використанні кадрової різниці за допомогою двох кадрів можуть виникати дублікати при певних кольорах цілі та фону, наприклад, при чорній цілі та білому фоні, або навпаки.

Розроблено алгоритм визначення координат рухомого об’єкту за даними двох відеокамер у прямокутній системі координат. При аналізі точносних характеристик отримано, що при приближенні рухомого об’єкта до лінії на якій розташована база датчиків, СКВ стрімко зростає (пеленг цілі наближається до 90° або 270°, в залежності з якого боку летить ціль), що створює так звані «сліпі» зону. Тому при побудові мережі відеодатчиків необхідно розташовувати бази таким чином, щоб сліпі зони в області спостереження не перетинались.

Використання калманівської фільтрації дозволяє визначити швидкість руху цілі з високою точністю (23.3 рази для осі X у , в 60.5 рази для осі Y, в 55.4 рази для осі Z), зменшити СКВ помилок оцінки та прогнозу. Для похибки прогнозу значення

СКВ було зменшено для осі X у 6.9 рази, для осі Y у 8 рази, для осі Z у 7.5 рази. Для похибки оцінки СКВ було зменшено для осі X у 3.2 рази, для осі Y у 3.3 рази, для осі Z у 3.2 рази.

При віддалені рухомої цілі від мережі відеокамер значення СКВ оцінювання параметрів руху цілі збільшується.

Для реалізації калманівської фільтрації для мережі із двох відеокмамер треба використовувати мікроконтролер STM32F205ZF. Під час обробки даних у реальному часі потрібно зробити 2304 операції, 1233 з яких – це множення, а 1071 -додавання) на один такт фільтрації. У такому випадку мінімальна кількість комірок пам’яті становить 183. При умові, що за секунду надсилаються камерами 30 кадрів, то орієнтовна максимальна кількість операцій за секунду становить 69120.

У випадку збільшення кількості камер є два можливі варіанти:

- Вертикальне масштабування — підбор мікроконтролера з більшими технічними характеристиками
- Горизонтальне масштабування — створення нових груп відеодатчиків у мережі, які складаються с однакових камер і однакових мікроконтролів, які оброблють дані у рамках однієї групи з двох датчиків.

Алгоритми калманівської фільтрації легко розповсюджується на багатоканальних систем, що дозволяє додавати відеокамери з іншими базами та потім об’єднувати отримані вимірювання у єдиний фінальний результат. Такий підхід дозволить створити більш детальну картину руху цілі, що сприятиме ефективнішому моніторингу траєкторії рухомої цілі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Deep learning-based strategies for the detection and tracking of drones using several cameras [Електронний ресурс] / P.Dupouy, E. Zenou, N. Riviere, E. Unlu. – 2019. – Режим доступу до ресурсу: https://ipsjcva.springeropen.com/articles/10.1186/s41074-019-0059-x.
2. Стандарти НАТО в безпілотній авіації [Електронний ресурс] – Режим доступу до ресурсу: https://www.ukrmilitary.com/2020/03/nato-uav-standart.html.
3. Кривоніс О. П. Системи виявлення дронів і протидронні системи [Електронний ресурс] / Олександр Петрович Кривоніс – Режим доступу до ресурсу: https://www.bezpeka-shop.com/ua/blog/obzor/sistemy-obnaruzheniya-dronov-i-protivodronnye-sistemy/.
4. OpenCV: Optical Flow [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html.
5. OpenCV: Meanshift and Camshift [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/3.4/d7/d00/tutorial_meanshift.html.
6. Математична морфологія [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikiwand.com/uk/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B0_%D0%BC%D0%BE%D1%80%D1%84%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F.
7. Structural Analysis and Shape Descriptors [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html.
8. Camera Calibration and 3D Reconstruction [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html.
9. Rotation matrix [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikiwand.com/en/Rotation_matrix.

10. Dastgheib M. A Step by Step Mathematical Derivation and Tutorial on Kalman Filters [Електронний ресурс] / M. Dastgheib, H. Masnadi-Shirazi, A. Masnadi-Shirazi. – 2019. – Режим доступу до ресурсу: https://arxiv.org/pdf/1910.03558.
11. Кот М. Г. Використання радіохвильових технологій для відстежування та розпізнавання людей у приміщенні з використанням нейронних мереж. V Всеукраїнської науково-технічної конференції студентів та аспірантів «Радіоелектроніка в XXI столітті» 10 – 12 травня 2023 Київ, Україна с.51-52.
12. Кот М.Г. Методика розрахунку розміру зображення БПЛА в пікселях в залежності від відстані до нього / М.Г. Кот, К.Ф. Соколов, Жук С.Я. // Міжнародна науково-технічна конференція «Радіотехнічні проблеми, сигнали, апарати та системи»: матеріали конференції, 14 грудня 2023 р., м. Київ, Україна / КПІ ім. Ігоря Сікорського, РТФ. – Київ : КПІ ім. Ігоря Сікорського, 2023.

ДОДАТОК А

```
import cv2
import numpy as np
import math
from PossibleTarget import PossibleTarget
from matplotlib import pyplot as plt

cap = cv2.VideoCapture("..\test-video.mp4")

count = 0
center_points_prev_frame = []

MAX_DISTANCE = 150

MEASURED_COORDS = []

tracking_objects = {}
track_id = 0

prev_frame = None

prev_cX = None
prev_cY = None

while True:
    ret, frame = cap.read()
    count += 1
    if not ret or count > 1200:
        break

    if count > 325:
        break

    center_points_cur_frame = []

    if prev_frame is None:
        prev_frame = frame
        continue

    curr_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY).astype(np.float32)
    prev_img = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY).astype(np.float32)

    diff = cv2.absdiff(curr_img, prev_img)

    max_d = np.max(diff)
    min_d = np.min(diff)

    binarization_limit = int(0.5 * (max_d - min_d))

    _, Q = cv2.threshold(diff, binarization_limit, 255, cv2.THRESH_BINARY)

    Q = Q.astype(np.uint8)

    Q = cv2.GaussianBlur(Q, (3, 3), cv2.BORDER_DEFAULT)
    Q = cv2.medianBlur(Q, 3)

    Q = cv2.dilate(Q, None, iterations=3)

    Q = cv2.GaussianBlur(Q, (3, 3), cv2.BORDER_DEFAULT)
    boxes, _ = cv2.findContours(Q, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for box in boxes:
        area = cv2.contourArea(box)
```

```
            (x, y, w, h) = cv2.boundingRect(box)
            M = cv2.moments(box)

            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])

            MEASURED_COORDS.append([cX, cY])

            possible_target = PossibleTarget(cX, cY)
            center_points_cur_frame.append(possible_target)

            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        tracking_objects_copy = tracking_objects.copy()
        center_points_cur_frame_copy = center_points_cur_frame.copy()
        for object_id, pt2 in tracking_objects_copy.items():
            object_exists = False
            for pt in center_points_cur_frame_copy:
                distance = math.hypot(pt2.x - pt.x, pt2.y - pt.y)

                if distance < MAX_DISTANCE:

                    existing_obj = tracking_objects.get(object_id)
                    existing_obj.update_coords(pt.x, pt.y)

                    tracking_objects[object_id] = existing_obj
                    object_exists = True
                    if pt in center_points_cur_frame:
                        center_points_cur_frame.remove(pt)
                    continue


        for pt in center_points_cur_frame:
            pt.center_history.append(((pt.x, pt.y), (pt.x, pt.y)))
            tracking_objects[track_id] = pt
            track_id += 1


        for object_id, pt in tracking_objects.items():
            obj_id = str(object_id)
            cv2.circle(frame, (pt.x, pt.y), 5, (0, 0, 255), -1)
            cv2.putText(frame, obj_id, (pt.x, pt.y - 7), 0, 1, (0, 0, 255), 2)


            for prev, curr in pt.center_history:
                cv2.line(frame, prev, curr, (255, 0, 0), 2)

            prev_cY = pt.y
            prev_cX = pt.x

        key = cv2.waitKey(30)
        if key == 27 or key == ord('q'):
            break

width = int(cap.get(3))
height = int(cap.get(4))

cap.release()
cv2.destroyAllWindows()
```