**РЕФЕРАТ**

**Актуальність теми.** Дослідження в напрямку системи технічного зору (СТЗ) та її інтегрування в систему керування безпілотного літального апарату (БПЛА) спрямовані на вирішення декількох найважливіших задач – стеження БПЛА за ціллю та його автоматична посадка на спеціальне підготовлене місце. СТЗ в даний час є одним з головних засобів розвитку автоматичних систем керування рухом в умовах, коли обсяг апріорної інформації не достатній і для вирішення завдань керування необхідний аналіз зовнішньої обстановки в режимі реального часу. Прогрес у електроніці, комп’ютерній техніці та програмуванні дає можливість використовувати оптичні прилади та системи у широкому спектрі приладів споживчого ринку. Проблемі дослідження використання СТЗ у авіації для навігації або наведення присвячено багато робіт, проте дане дослідження не втрачає своєї актуальності, оскільки нерозв’язних питань доволі багато.

Суттєвою проблемою використання СТЗ в керуванні БПЛА є його синхронізація органів керування (гвинти-двигуни) з комп’ютерним зором.

Наступна нерозв’язана задача залишається в оптимізації зображення місцевості, оскільки на СТЗ дуже суттєво впливає зміна стану навколишнього середовища (освітленість, яскравість, інтенсивність). Алгоритми ідентифікації об’єктів є дуже чутливими до цих параметрів, а при неоптимальній обробці зображення можуть виникати шуми на зображенні. Також проблемою є підбір необхідних фільтрів шумів, оскільки неправильна фільтрація може призвести до втрати важливої інформації з зображення місцевості, що в свою чергу може призвести до нездатності знаходження цікавлячих об’єктів.

Проблема надійності даних систем: використання одного методу ідентифікації цілі є не завжди надійним, тому краще комбінувати різні методи, особливо різні за їх природою розпізнавання. Більшість проектів використовують тільки один метод розпізнавання.

Повністю невирішена задача посадки БПЛА типу квадроптер на малій висоті на основі СТЗ без використання інерційних датчиків/висотомірів, що суттєво зменшує точність посадки в існуючих алгоритмах керування БПЛА при його знижені.

Також проблематикою залишається вибір відповідного алгоритму ідентифікації цілі та саму ціль (його параметри та особливості), виходячи з поставлених задач, оскільки вони не є універсальні для кожного БПЛА та характеристик місцевості.

Для вирішення даних проблем в даній роботі розроблено алгоритм автоматичної посадки БПЛА типу квадрокоптер на основі СТЗ на спеціально визначене місце, яке має свою особливу символіку (символ і певні позначки).

Пропонується комбінований спосіб (який не залежить від глибини зображення для уникнення зайвих шумів) ідентифікації місця посадки, оснований на декільком методів розпізнавання як символу, так і певних особливостей/ознак майданчику посадки, для підвищення надійності даного алгоритму, а також введення в систему керування БПЛА закону виконання стабілізації в непередбачуваних ситуаціях (при втраті зображення чи цілі).

Підбираються та вводяться нові методи фільтрації для оптимізації та покращення зображення, максимальне видалення шумів. Комбінація даних фільтрів з методами розпізнавання місця посадки є особливим в даній роботі, оскільки йде підбір найбільш оптимального знаходження місця посадки в умовах всепогодності навколишнього середовища при умові, що є освітленість.

Вирішується проблема посадки на малих висотах за методом «target in target» (новий метод: розміщення об’єкту розпізнавання в самій символіці місця посадки, який можливо побачити тільки з малих відстаней, а також новий багатомасштабований алгоритм розпізнавання символу). Очікується точність посадки між центром майданчика та вертикальної осі БПЛА. Саме тому дана робота є **актуальною** науково-дослідною роботою.

**Наукова новизна** дисертації полягає у розробці алгоритмів автоматичного керування БПЛА на основі інформації з СТЗ та методів інтеграції СТЗ в систему керування БПЛА типу квадрокоптер.

1) Вперше запропоновано багатомасштабований алгоритм розпізнавання орієнтира за методом «target in target» для вирішення проблеми посадки БПЛА на малих висотах.

2) Вдосконалено метод комплексування алгоритмів ідентифікації орієнтира, які мають різні принципи роботи.

3) Розвинено метод використання нових фільтрів з максимальним видаленням шумів для оптимізації зображення місцевості з використанням фільтру Калмана.

**Практичне значення** отриманих результатів визначається вимогами, які висуваються до системи об’єкта, на якому буде розташовуватися комплексна система. Такий підхід дозволить визначити доцільність використання комплексної системи на об’єкті та модернізувати алгоритм обробки інформації.

**Ключові слова**. СИСТЕМА ТЕХНІЧНОГО ЗОРУ, ОПТИКО – ЕЛЕКТРОННА СИСТЕМА, СИСТЕМА КЕРУВАННЯ БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТУ, ФІЛЬТР КАЛМАНА.

**РЕФЕРАТ**

**Актуальность темы.** Исследования в направлении системы технического зрения (СТЗ) и ее интегрирования в систему управления беспилотного летательного аппарата (БПЛА) направлены на решение нескольких важных задач - наблюдение БПЛА за целью и его автоматическая посадка на специальное подготовленное место. СТЗ в настоящее время является одним из главных средств развития автоматических систем управления движением в условиях, когда объем априорной информации не достаточен и для решения задач управления необходим анализ внешней обстановки в режиме реального времени. Прогресс в электронике, компьютерной технике и программировании дает возможность использовать оптические приборы и системы в широком спектре устройств потребительского рынка. Проблема исследования использования СТЗ в авиации для навигации или наведения посвящено много работ, однако данное исследование не теряет своей актуальности, поскольку неразрешимых вопросов достаточно много.

Существенной проблемой использования СТЗ в управлении БПЛА является его синхронизация органов управления (винты-двигатели) с компьютерным зрением.

Следующая нерешенная задача остается в оптимизации изображения местности, поскольку на СТЗ очень существенно влияет изменение состояния окружающей среды (освещенность, яркость, интенсивность). Алгоритмы идентификации объектов являются очень чувствительными к этим параметрам, а при неоптимальной обработке изображения могут возникать шумы на изображении. Также проблемой является подбор необходимых фильтров шумов, поскольку неправильная фильтрация может привести к потере важной информации из изображения местности, в свою очередь может привести к неспособности нахождения интересующих объектов.

Проблема надежности данных систем: использование одного метода идентификации цели является не всегда надежным, поэтому лучше комбинировать различные методы, особенно разные по их природе распознавания. Большинство проектов используют только один метод распознавания.

Полностью нерешенная задача посадки БПЛА на малой высоте на основе СТЗ без использования инерционных датчиков / высотомеров, что существенно уменьшает точность посадки в существующих алгоритмах управления БПЛА при его снижены.

Также проблематикой остается выбор соответствующего алгоритма идентификации цели и саму цель (его параметры и особенности), исходя из поставленных задач, поскольку они не являются универсальными для каждого БПЛА и характеристик местности.

Для решения данных проблем в данной работе разработан алгоритм автоматической посадки БПЛА типа квадрокоптер на основе СТЗ на специально определенное место, которое имеет свою особую символику (символ и определенные отметки).

Предлагается комбинированный способ (не зависит от глубины изображения во избежание лишних шумов) идентификации места посадки, основанный на нескольких методов распознавания как символа, так и определенных особенностей / признаков площадки посадки, для повышения надежности данного алгоритма, а также введение в систему управления БПЛА закона выполнения стабилизации в непредвиденных ситуациях (при потере изображения или цели).

Подбираются и вводятся новые методы фильтрации для оптимизации и улучшения изображения, максимальное удаление шумов. Комбинация данных фильтров с методами распознавания места посадки является в данной работе, поскольку идет подбор наиболее оптимального нахождения места посадки в условиях всепогодности окружающей среды при условии, что есть освещенность.

Решается проблема посадки на малых высотах по методу «target in target» (новый метод: размещение объекта распознавания в самой символике места посадки, который можно увидеть только с малых расстояний, а также новый багатомасштабований алгоритм распознавания символа). Ожидается точность посадки ± 5см между центром площадки и вертикальной оси БПЛА. Именно поэтому данная работа является **актуальной** научно-исследовательской работой.

**Научная новизна** диссертации состоит в разработке алгоритмов автоматического управления БПЛА на основе информации из СТЗ и методик интеграции СТЗ в систему управления БПЛА типу квадрокоптер.

1) Впервые предложено багатомасштабований алгоритм распознавания ориентира по методу «цель в цели» для решения проблемы посадки БПЛА на малых высотах.

2) Усовершенствован метод комплексирования алгоритмов идентификации ориентира, которые имеют различные принципы работы.

3) Развита метод использования новых фильтров с максимальным удалением шумов для оптимизации изображения местности с использованием фильтра Калмана.

**Практическое значение** полученных результатов определяется требованиям, предъявляемым к системе объекта, на котором будет располагаться комплексная система. Такой подход позволит определить целесообразность использования комплексной системы на объекте и модернизировать алгоритм обработки информации.

**Ключевые слова.** СИСТЕМА ТЕХНИЧЕСКОГО ЗРЕНИЯ, ОПТИКО - ЭЛЕКТРОННАЯ СИСТЕМА, СИСТЕМА УПРАВЛЕНИЯ БЕСПИЛОТНЫХ ЛЕТАТЕЛЬНЫХ АППАРАТОВ, ФИЛЬТРА КАЛМАНА.

**ABSTRACT**

**Actuality of theme.** Research in the direction of the technical vision system (TVS) and its integration into the control of an unmanned aerial vehicle (UAV) are aimed at solving several of the most important tasks - tracking the UAV for the purpose and automatic landing on a specially prepared place. TVS is currently one of the main means of developing automatic traffic control systems in conditions where the volume of a priori information is not sufficient and to solve the problems of management requires an analysis of the external environment in a real time. Progress in electronics, optics, computer technology and programming makes it possible to use optical devices and systems in a wide range of consumer market devices. The problem of researching the use of TVS in aviation for navigating or guiding is devoted to many works, but this study does not lose its relevance because of insoluble issues quite a lot.

The essential problem of using TVS in UAV control is its synchronization of controls (screw engines) with computer vision.

The next unresolved problem remains in optimizing the image of the terrain, because the change in the state of the environment (illumination, brightness, intensity) is very significant on TVS. Algorithms for object‘s identification are very sensitive to these parameters, and when non-optimal image processing, there may be noise in the image. The problem is also the selection of the required noise filters, because improper filtration can lead to the loss of important information from the image of the terrain, which in turn can lead to the inability to find interesting objects.

The problem of reliability of these systems: the use of one goal identification method is not always reliable, so it is better to combine different methods, especially different in their nature of recognition. Most projects use only one recognition method.

Completely unsolved problem of UAV landing at low altitude based on TVS without the use of inertial sensors / altimeter, which significantly reduces the accuracy of landing in existing UAV control algorithms when it is lowered.

Also, the choice of the appropriate algorithm for identifying the target and the same objective (its parameters and features) remains the problem, based on the tasks set, since they are not universal for each UAV and the characteristics of the area.

To solve these problems in this work an algorithm of automatic landing of UAVs of a type of kvadokopter on the basis of a TVS on a specially defined place, which has its own special symbolism (a symbol and certain marks) is developed.

The combined method (which does not depend on the depth of the image to avoid excessive noise) is proposed, identification of the landing site based on several methods of recognizing both the symbol and certain features / attributes of the landing site, in order to increase the reliability of the algorithm, as well as the introduction into the control circuit of the UAV by law performance of stabilization in unforeseen situations (with loss of image or purpose).

Selection and introduction of new filtering techniques to optimize and improve the image, maximize noise removal. The combination of filter data with the methods of recognizing the landing site is special in this work, because the selection of the most optimal location of landing in conditions of all-weather environment, provided that there is illumination.

The problem of landing at low altitudes is solved using «target in target» method (a new method: the placement of the recognition object in the symbolism of the landing site, which can only be seen from small distances, as well as a new multi-scale algorithm for character recognition). The accuracy of landing is expected to be ± 5 cm between the center of the site and the vertical axis of the UAV. That is why this work is an actual research work.

**The scientific novelty** of the dissertation is to develop algorithms for automatic control of UAVs on the basis of information from the STZ and techniques for integrating the STZ into the control circuit of the UAV.

1) For the first time, a multi-scale algorithm for recognizing the landmark in the "goal to the target" method was proposed for solving the problem of UAV landing at low altitudes.

2) The method of integration of orientation identification algorithms, which have different principles of work, is improved.

3) The method of using new filters with the maximum noise removal for the optimization of the terrain image using the Kalman filter is developed.

The practical value of the results obtained is determined by the requirements put forward to the system of the object, which will be located on the complex system. Such an approach will allow us to determine the feasibility of using an integrated system on an object and modernize the algorithm of information processing.

**Keywords.** SYSTEM OF TECHNICAL VISION, OPTICAL - ELECTRONIC SYSTEM, CONTROL SYSTEM OF SAFE LITERAL APPARATUS, FILTER KALMAN.

**ЗМІСТ**

Перелік скорочень та умовних позначень…………………………………………..18

ВСТУП………………………………………………………………………….……..19

1. ОГЛЯД ЛІТЕРАТУРИ……………………………………………………………..24

1.1. Аналіз сучасного стану використання систем технічного зору в безпілотних літальних апаратах………………………………………………………………...….24

1.2. Аналіз сучасних методів вирішення автоматичної посадки БПЛА на основі СТЗ…………………………………………………………………………….……….24

1.2.1. Системи посадки на основі технічного зору…………………………….……26

1.2.1.1. Контроль / моніторинг для посадки безпілотних літальних апаратів….…26

1.2.1.2. Слідкуючі системи технічного зору в контурі посадки безпілотних літальних апаратів………………………………………………………………….…29

1.2.2. Методи виявлення кутів в алгоритмах посадки БПЛА…………………..….33

1.3. Розгляд найближчих аналогів проекту…………………………………….……34

1.3.1. Автоматична посадка на рухому мішень для безпілотних вертольотів на основі технічного зору……………………………………………………………..…34

1.3.2. Автономна навігація вертольотного безпілотного літального апарату з використанням машинного зору…………………………………………………..…35

1.4. Розв’язані питання в літературних джерелах…………………………………..35

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ…………………………………………..…36

2. АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМИ ТЕХНІЧНОГО ЗОРУ……………………39

2.1. Оптико-електронні системи в СТЗ………………………………………………40

2.1.1. Роздільна здатність оптико-електронних систем………………………….…41

2.1.2. Методика вибору характеристик оптичної системи…………………………42

2.2. Обчислювальні системи безпілотних літальних апаратів як частина системи технічного зору………………………………………………………………………46

2.2.1. Архітектура бортових обчислювальних систем………………………….…46

2.2.2. Операційні системи бортової авіоніки безпілотних літальних апаратів…..47

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ…………………………………………..…48

3. ВИБІР СКЛАДОВИХ СИСТЕМИ ТЕХНІЧНОГО ЗОРУ………….………...…50

3.1. Вибір оптичної системи….……………………………………......……………50

3.2. Вибір електронної обчислювальної машини……………………………….…54

3.2.1. Вибір електронної обчислювальної машини для БПЛА………………........55

3.2.2. Вибір електронної обчислювальної машини для лабораторного стенда….57

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ……………………………………………58

4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТЕХНІЧНОГО ЗОРУ………..…59

4.1. Підключення камери і настройка передачі зображення…………………...…60

4.2. Розпізнавання зображення – система комп’ютерного зору…………….…….61

4.2.1. Метод ідентифікації об’єкта за його кольором……………………….…..…62

4.2.1.1. Загальний огляд методу…………………………………………….…….…62

4.2.1.2. Розробка програмного забезпечення алгоритму ідентифікації об’єкта за його кольором……………………………………………………………….…..……66

4.2.2. Метод ідентифікації об’єкта за особливими точками…………….……....…66

4.2.2.1. Загальний огляд методу………………………………………….……….…66

4.2.2.2. Розробка програмного забезпечення алгоритму ідентифікації об’єкта за особливими точками…………………………………………………………….……71

4.3. Застосування фільтра Калмана в системі технічного зору з урахуванням динаміки руху БПЛА……………………………………………………………....…76

4.3.1. Синтез фільтра Калмана комплексної системи орієнтації та навігації…..…77

4.3.2. Реалізація фільтру Калмана………………………………………………....…80

ВИСНОВКИ ДО ЧЕТВЕРТОГО РОЗДІЛУ……………………………………....…82

5. МОДЕЛЮВАННЯ ДИНАМІКИ ПОЛЬОТУ БПЛА ТИПУ КВАДРОКОПТЕР..84

5.1. Розробка і аналіз рівнянь квадроторної системи…………………………….…84

5.1.1. Змінні та координатне визначення системи………………………………..…84

5.1.2. Кінематика переходу……………………………………………………...……86

5.1.3. Кінематика обертального руху……………………………………………...…87

5.1.4. Рівняння руху безщіточного двигуна…………………………………………88

5.1.5. Динаміка лінійного руху…………………………………………………….…90

5.1.6. Динаміка обертального руху…………………………………………………92

5.1.7. Рівняння руху…………………………………………………………….……93

5.2. Оцінка параметрів моделі, основані на нашій фізичній системі…………..…94

5.2.1. Параметри фізичної моделі………………………………………………...…94

5.2.1.1. Момент інерції квадрокоптера…………………………………………...…94

5.2.1.2. Момент інерції пропелера…………………………………………………...96

5.2.2. Характеристика датчиків…………………………………………………....…96

5.2.2.1. Моделювання роботи гіроскопа…………………………………………..…97

5.2.2.2. Моделювання акселерометра……………………………………………..…99

5.2.3. Параметри системи двигуна……………………………………………….…101

5.2.3.1. Ідентифікація динаміки двигуна………………………………………...…102

5.2.3.2. Коефіцієнт тяги…………………………………………………………...…106

5.2.3.3. Максимальна швидкість двигуна…………………………………………..107

5.3. Розробка системи керування………………………………………………...…108

5.3.1. Загальна стратегія керування……………………………………………...…108

5.3.2. Спрощені рівняння руху…………………………………………………...…110

5.3.3. Спрощений ПІД-контролер………………………………………………..…112

5.3.4. Розробка системи керування…………………………………………………114

5.3.4.1. Контур керування позиції квадрокоптера в глобальній СК…………...…115

5.3.4.2. Контур керування орієнтації та висоти……………………………………116

5.3.4.3. Контур керування кутовою швидкістю……………………………………117

5.3.4.4. Керування двигуном……………………………………………………...…117

5.3.4.5. Основні модифікації під – «похідний удар» …………………………...…119

5.3.4.6. Основні модифікації під - anti-windup………………………………..……120

5.4. Симуляція динаміки польоту квадрокоптера в програмному забезпечені Matlab…………………………………………………………………………………121

ВИСНОВКИ ДО П‘ЯТОГО РОЗДІЛУ………………………………………..……136

6. АВТОМАТИЧНА ПОСАДКА КВАДРОКОПТЕРА НА ОСНОВІ СИСТЕМИ ТЕХНІЧНОГО ЗОРУ……………………………………………………………..…137

6.1. Алгоритм роботи наведення квадрокоптера……………………………..……137

6.1.1. Симуляція наведення квадрокоптера за даними системи технічного зору в програмному забезпечені Matlab.………………………………………………..…139

6.2. Алгоритм роботи посадки квадрокоптера…………….……………………….140

6.2.1. Траєкторія посадки квадрокоптера………………………………………..…141

6.2.1.1. Посадка на малій висоті з урахуванням характеристик системи технічного зору……………………………………………………………………………………142

6.2.2. Симуляція автоматичної посадки квадрокоптера за даними системи технічного зору в програмному забезпечені Matlab………………………………144

ВИСНОВКИ ДО ШОСТОГО РОЗДІЛУ………………………………………...…145

7. РОЗРОБЛЕННЯ СТАРАП-ПРОЕКТУ………………………………………..…147

7.1. Опис ідеї проекту……………………………………………………………….147

7.1.1. Зміст ідеї…………………………………………………………………….…147

7.1.2. Аналіз потенційних техніко-економічних переваг ідеї………………….…147

7.2. Технологічний аудит ідеї проекту……………………………………………..149

7.3. Аналіз ринкових можливостей запуску стартап-проекту……………………149

7.3.1. Аналіз попиту…………………………………………………………………149

7.3.2. Потенційні групи клієнтів……………………………………………………150

7.3.3. Аналіз ринкового середовища…………………………………………….…151

7.3.4. Аналіз пропозиції……………………………………………………………..152

7.3.5. Детальний аналіз умов конкуренції в галузі……………………………..…152

7.3.6. Визначення та обгрунтування факторів конкурентоспроможності…….…153

7.3.7. Аналіз сильних та слабких сторін старап проекту………………………….153

7.3.8. Складання SWOT-аналізу………………………………………………….…154

7.3.9. Альтернативи ринкової поведінки………………………………………...…154

7.4. Розроблення ринкової стратегії проекту………………………………………156

7.4.1. Визначення стратегій охоплення ринку…………………………………..…156

7.4.2. Базова стратегія розвитку………………………………………………….…156

7.4.3. Стратегія Конкурентної поведінки………………………………………..…158

7.4.4. Стратегія позиціонування………………………………………………….…158

7.5. Розроблення маркетингової програми стартап-проекту……………………...159

7.5.1. Маркетингова концепція товару…………………………………………..…159

7.5.2. Трирівнева маркетингова модель товару……………………………………159

7.5.3. Визначення цінових меж…………………………………………………...…160

7.5.4. Оптимальна система збуту………………………………………………...…161

7.5.5. Розроблення концепції маркетингових комунікацій…………………….…161

ВИСНОВКИ ДО СЬОМОГО РОЗДІЛУ……………………………………………162

ВИСНОВКИ ПО РОБОТІ………………………………………………………...…164

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А. Вихідний код Python для здійснення відеопотоку

ДОДАТОК Б. Вихідний код Python реалізації алгоритму розпізнавання об’єкту за кольором

ДОДАТОК В. Вихідний код С++ реалізації ідентифікації об’єкта за особливими точками

ДОДАТОК Г. Скрипт Matlab для аналізу даних передатної функції двигуна

ДОДАТОК Ґ. Блок-схема взаємозв’язоку контролерів із системою керування квадрокоптера

ДОДАТОК Д. Вихідний код Matlab застосування фільтру Калмана для системи технічного зору

ДОДАТОК Е. Вихідний код Matlab реалізації алгоритму побудови точок траєкторії посадки квадрокоптера

**ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ**

СТЗ/СМЗ – система технічного/машинного зору

СКЗ – система комп’ютерного зору

ЛА – літальний апарат

БПЛА – безпілотний літальний апарат

СК – система координат

ПК – портативний комп’ютер

ТЗ – технічне завдання

ЦСП – цифровий сигнальний процесор

ІЧ – інфрачервоний

ПЗЗ – прилади із зарядовим зв’язком

ТВ – телевізійний

ЕОМ – електронна обчислювальна машина

КБО – комплекс бортового обладнання

ЦОС – цифрова обробка сигналів

БОС – бортова обчислювальна система

ІМА – інтегрована модульна авіоніка

САК – система автоматичного керування

РМ – рульовий механізм

МК - мікрокомп’ютер

ОП/ОД – оптичний пристрій/датчик

ПЗ – програмне забезпечення

ОС – операційна система

ФПС – кадри за секунду

СКЛА / СУЛА – система керування / управління літального апарату

ПІД / PID – пропорціний-інтегральний-диференційний

ШІМ / PWM – широтно-імпульсна модуляція

**ВСТУП**

Системи керування сучасними літальними апаратами (ЛА) призначені для керування складними багатофункціональними об'єктами, чинними в складній навколишній обстановці. При цьому канал зорового сприйняття є одним з найбільш важливих джерел інформації як у автоматичних, так і автоматизованих (людино-машинних) системах керування. Внаслідок цього в останні роки на передній план все більшою мірою виходять задачі створення систем технічного зору (СТЗ) для різних типів ЛА.

Системи технічного зору застосовуються для обробки і розпізнавання різних зображень, отриманих з телекамери або фотоапарата та є засобом спостереження і автоматичного прийняття рішення в тих чи інших випадках.

СТЗ знаходять своє застосування в сучасних космічних, авіаційних, наземних, надводних і підводних мобільних об'єктах. Завдяки дальньої дії достатньо високих рівнів просторового і колірного дозволу сучасних лінійних і матричних приймачів оптичного випромінювання СТЗ можуть служити незамінними джерелами інформації при автоматичному вирішенні завдань розпізнавання, навігації або наведення.

Разом з тим слід визнати наявність технічних труднощів впровадження зорового інформаційного каналу в контури систем керування літальними об'єктами. Зазначені труднощі пов'язані з проблемами перетворення зорової інформації в дані результатів розпізнавання, навігаційної прив'язки або визначення параметрів руху цих об'єктів в режимі реального часу.

Сьогодні мова йде про найбільш ефективні способи побудови та використання цілісної системи керування, зв'язку, обробки та збору інформації (так звані системи С4I). Ця система включає безліч ЛА, а також цілий ряд підсистем, служб і мереж, у тому числі пов'язаних із зображеннями та географічними даними. У неї входять космічні та авіаційні платформи отримання даних, служби географічної підтримки, наземні системи збору інформації, системи планування операцій, моделювання, навігації, керування рухом, цілевказівки і ряд інших. Невід'ємною частиною побудови системи С4I є збір і комплексування інформації, що надходить від датчиків різної фізичної природи, таких як космознімки, авіознімки, відео- та ІЧ-дані, дані від радарів, лазерних систем і т. д. З урахуванням цього літак початку XXI ст. повинен забезпечувати швидку і стійку передачу великих обсягів даних, високу точність навігаційних систем, надвелику дальність застосування, всепогодність, високу точність і гнучкість при виконанні поставлених завдань. Створення та застосування різних технологій аналізу зображень вимагають сьогодні системи збору інформації на базі космічних, авіаційних і БПЛА носіїв, системи підготовки польотних завдань, системи високоточної геопросторової прив'язки, системи автономного керування високоточних безпілотних ЛА і літаючих робототехнічних комплексів [1].

Тому актуальними є задачі створення систем технічного зору (СТЗ). Вони вирішуються різними методами: ідентифікацією об’єктів, яка полягає у тому, щоб вирізнити певний конкретний об’єкт серед йому подібних [2], віднесенням об’єкта до того чи іншого класу [2], кластерним аналізом, який полягає в розділенні заданого набору об’єктів на класи – групи схожих між собою об’єктів за певним критерієм [2]. Вибір та ефективність використання методу значним чином залежить від конкретних випадків.

**Мета** дисертаційної роботи полягає в розробці системи автоматичного керування безпілотного літального апарату за допомогою системи технічного зору для вирішення задачі його автоматичної посадки на спеціально підготовлену площадку в реальному масштабі часу з заданою точністю.

**Об’єктом дослідження** є система технічного зору для системи керування БПЛА типу квадрокоптер.

**Предметом дослідження** є сучасні методи розпізнавання образів для формування їх як навігаційних орієнтирів і використання в системі керування БПЛА на основі інформації від СТЗ.

**Методи дослідження.** Опис принципів побудови оптико-електроннох системи; огляд сучасних існуючих методів розпізнавання орієнтирів оптико-електронної системи; розробка системи компютерного зору; опис принципів керування БПЛА типу квадрокоптер; розробка комбінованої системи атоматичного керування БПЛА на основі СТЗ.

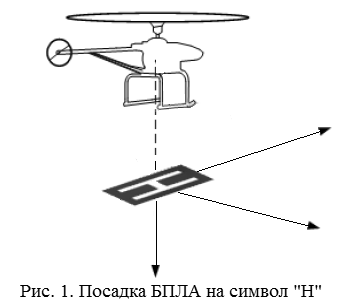
**Наукова новизна** дисертації полягає у розробці алгоритмів автоматичного керування БПЛА типу квадрокоптер на основі інформації з СТЗ та методів інтеграції СТЗ в систему керування БПЛА типу квадрокоптер.

1) Вперше запропоновано багатомасштабований алгоритм розпізнавання орієнтира за методом «target in target» для вирішення проблеми посадки БПЛА на малих висотах.

2) Вдосконалено метод комплексування алгоритмів ідентифікації орієнтира, які мають різні принципи роботи.

3) Розвинено метод використання нових фільтрів з максимальним видаленням шумів для оптимізації зображення місцевості з використанням фільтру Калмана.

Формалізація задачі: на вхід подається зображення сцени із камери і зображення цільового плоского об'єкта (наприклад символ «Н»). Завдання - знайти цільовий об'єкт на зображенні сцени, його точні координати відносно зображення сцени, і видати значення цих координат на систему керування для здійснення наведення БПЛА (керування двигунів БПЛА) на ціль для стабілізації та посадки.



Дана система технічного зору може знайти практичне застосування в розвідувальних операціях, відтворювання карт місцевості, спостереження території, транспортування різних посилок. Наприклад, нехай ми відправляємо наш БПЛА з точки А до точки Б. По GPS навігатору ми даємо координати точки Б. В автономному режимі він здійснює переліт. Опинившись на координатах кінцевої точки його траєкторії, він виконує автоматичну посадку на вказане місце (символ «Н») без втручання людини. Тобто людині не потрібно керувати політ БПЛА, що дає їй змогу займатися іншим.

Також БПЛА з СТЗ може виконувати військові операції (розвідка в тилу противника), оскільки є повністю автономним, але потрібно буде доповнити цей СТЗ алгоритмом повернення БПЛА до точки старту (в цій точці також може знаходитися символ «Н» для автоматичної посадки) наприклад по знімках, зроблених відеокамерою, у разі втрати сигналу від GPS, що забезпечить збереженню військової таємниці й затрачених на цей ЛА ресурсів.

Магістерська дисертація «Автоматична посадка беспілотного літального апарату на основі системи технічного зору» містить 164 сторінки тексту, 50 ілюстрацій, 22 таблиці, 84 бібліографічних посилань.

**Публікації.** Зарезультатами досліджень опубліковано7 друковані роботи, з яких 3 – в статті в фахових журналах, 4 – тези доповідей на конференціях:

- Метод визначення координат рухомого об`єкту з використанням системи технічного зору /Котвицький Р.С., Збруцький О.В., Сарибога Г.В. // Інформаційні системи, механіка та керування.-2017.-№16. С. 71-78

- Автоматичне керування оптичною віссю камери на основі системи технічного зору з використанням методу ідентифікації об’єктів за кольором/Котвицький Р.С., Збруцький О.В., Сарибога Г.В. // Інформаційні системи, механіка та керування.-2015.-№13. С. 111-115

- Назва заходу XІ Международная научно-техническая конференция «Гиротехнологии, навигация, rерування движением и конструирование авиационно-космической техники»; Назва доповіді - Метод визначення координат рухомого об`єкту за допомогою системи технічного зору; Автори - КОТВИЦЬКИЙ Р.С., САРИБОГА Г.В., ЗБРУЦЬКИЙ О.В.; Місце проведення - Київ, КПИ им. Игоря Сикорского; Дата проведення: 13.04.2017

- VIІI науково-технічних конференціях студентів та молодих учених «Гіротехнології. Навігація. Керування рухом та конструювання авіаційно-космічної техніки» 2015;

- IХ міжнародній конференції студентів та молодих вчених «Intelligence. Integration. Reliability » 26 квітня 2016;

- науково-технічна конференція студентів та молодих учених «Гіротехнології, навігація, керування рухом та конструювання авіаційно-космічної техніки» 21.03.2018р.

1. **ОГЛЯД ЛІТЕРАТУРИ**
   1. **Аналіз сучасного стану  використання систем технічного зору в безпілотних літальних апаратах**

На даний час багато публікацій присвячено проблемним питанням процесу обробки цифрових фотозображень апаратурою безпілотного літального апарату. Основні з яких досліджуються в таких публікаціях: у науковій статті [2] розглядаються методи та алгоритми цифрової обробки в оптико-електронних приладах; у науковій статті [4] наведені стандарти передачі відео- та фотоінформації; у змістовній роботі [5] розглядаються вимоги до каналів зв’язку з БПЛА та визначені найбільш перспективні шляхи їх реалізації.

Цифрова обробка зображень отриманих з БПЛА набуває особливого поширення. Різноманітність методів та алгоритмів пов’язано з широким колом проблем, які виникають під час обробки та передачі цифрових даних в апаратурі БПЛА, а особливо проблеми обробки в реальному масштабі часу. На підставі проведеного аналізу тенденцій розвитку використання матриць ПЗЗ та алгоритмів цифрової обробки для отримання інформації з БПЛА [4–6], можливо зробити висновок, що на сьогодні є актуальним завданням:

– обґрунтування вимог до оптичної системи та процесу обробки цифрових фотозображень апаратурою БПЛА;

– вирішення протиріччя між отриманням необхідної якості зображень та забезпеченням максимально можливого стиснення даних.

* 1. **Аналіз сучасних методів вирішення автоматичної посадки БПЛА на основі СТЗ**

Обробка зображень низького рівня в системі бачення для використання на гвинтокрилих квадрокоптерах була розроблена Сильвестром в [7]. Він вибирає довільне місце посадки, виконаний з концентричних кільцевих візерунків і проводить аналіз зображення з використанням різних методів сегментації розрізнення майданчика посадки від фону. Він досяг стабільно хороших результатів у фільтрації зображень для видалення шуму, маркування зв’язних компонентів, а також виділення контурів для посадки на ціль з концентричними кільцевими візерунками. Але його результати є не корисними для посадки на ціль, основа якої є символ «H». Іншими недоліками є те, що не враховується технічне забезпечення, особливо підбір апаратних засобів та інтеграції їх в системі; а також зображення в основному не розглядаються і є нечутливими до обертання, розмірів та перемасштабуванню. Оскільки масштаб цієї роботи включає в себе оцінку стану БПЛА на основі посадки цільового місця розташування, робота Сильвестра є обмеженою.

Мбаякуб [8] працював над керуванням квадрокоптера і системи керування розвитку польоту шляхом складання стану-рельєфу (off-the-shelf) arducopter на основі квадроторного БПЛА, динаміка якого була змодельована з використанням ПІД-регулятора. Зауважимо, що його система стабілізується лише в певних межах робочих обмежень. БПЛА не літав через неточно обраний струм і напругу, відмову льотного контролера від зовнішнього збурення, а також ЛА мав високу чутливість до порушення орієнтації під час умов парування.

Таким чином, ця робота розділена на два етапи: по-перше, розробка повного літального апарату з дистанційним керуванням (RC) зльоту і посадки, з телеметрією на основі команди в польоті, вказувати висоту стабілізації щодо цілі; і по-друге, розробка алгоритм посадки, який включає розробку алгоритму комп'ютерного зору і подальшої інтеграції з системою БПЛА в структурі зворотного зв'язку для контролю посадки.

У міру того як основна частина цього проекту направлена на технічний зір на основі автономної посадки гвинтокрилого безпілотного літального апарату, цей огляд літератури зосереджений на дослідженні комп'ютерного зору, який існує в літературі.

* + 1. **Системи посадки на основі технічного зору**

Декілька робіт прагнули розробити, впровадити та інтегрувати системи технічного зору (далі СТЗ) в систему керування БПЛА типу квадрокоптер для здійснення автоматичної посадки. Розглядаються дві широкі категорії а саме, (а) систем контролю / стеження [9], [10], [11] і [12] є ті, які використовують системи бачення для виконання обробки зображень, сегментації, виділення особливих точок, оцінки руху для подальшого розроблення алгоритмів, які дозволяють оцінити стан БПЛА по відношенню до цілі посадки - але вони не впроваджують СТЗ в архітектуру контуру керування для автономної посадки; і (б) СТЗ стеження [13], [14], [15], [9], [16], [17] і [18], як правило, слідкуючи підходи, що і (а) категорії стеження, але комп'ютерний зір не інтегрований в структуру контуру зворотного зв'язку для автономної посадки. В даній роботі є автоматизованим планування місій, розпізнавання об'єктів, а також їх оцінювання.

* + - 1. **Контроль / моніторинг для посадки безпілотних літальних апаратів**

Датчик бачення (камера) в контурі зворотного зв'язку безпілотних літальних апаратів є привабливою особливістю, оскільки інерціальні датчики супроводжуються дрейфом з плином часу. Це відбувається тому, що позиційні вимірювання інтегруються по даним про швидкість, що призводять до шуму і неправильної оцінки стану або швидкості. Комп'ютерний зір апаратних засобів в цілому споживають менше енергії, менші за розміром; розрахункова вартість реалізації також є відносно дешевим для візуальної обробки даних, що робить вагомі аргументи на користь їх інтеграції в структуру контуру зворотного зв'язку.

Shakernia та ін. в [19] зробили важливий аналіз з використанням комп'ютерного зору в якості датчика зворотного зв'язку в контурі керування для посадки БПЛА, де проблеми СТЗ розглядалося як проблема самооцінки (ego-estimation problem), тобто "всі особливі точки лежать на плоскій поверхні". Єдина геометрична структура і схема оцінки представлені для вирішення диференціала. Ці алгоритми використовуються для реалізації комп'ютерного бачення в контурі зворотного зв'язку для оцінки стану контролю посадки. Алгоритми лінійні і здатні працювати в режимі реального часу навколишнього середовища, але вони фізично не продемонстрували цього. Експериментальні результати цього підходу пізніше опубліковані в [20]. Алгоритм, розроблений в [19], був ретельно вивчений з використанням різних критеріїв ефективності, таких як різні рівні шумів зображення, висоті камери над майданчиком посадки, а також контролер розроблений в рамках всеохоплюючого динамічної поведінки моделі безпілотного літального апарату.

Зорова система реалізована в [9] включає в себе замовне програмне забезпечення Java на основі графічного інтерфейсу візуалізації з зазначенням оцінок позиції і обертання БПЛА, що виділяють характерні точки і мітки, а також двійковий вид зображення камери. Піддослідний є вертоліт Берклі БЛА Yamaha R50, що складається зі змонтованої камери поворот/нахил і мікрокомп’ютера з бортовою навігацією і комп'ютерів виконання обробки зображень, сегментації і виділення ознак, керування камерою, а також з лінійних і нелінійних оптимізацій для оцінки в режимі реального часу. Тестові польоти показують точність в межах 5 см і 5 градусів в поступальному і обертальному русі осей відповідно, коли БЛА автономно завис над маленьким стаціонарним посадковим майданчиком. Система бачення не впливає на керування БЛА під час вимірювання INS / GPS, які показали точність в 2см. В цілому, оцінки СТЗ є досить корисними для розміщення в контурі керування ієрархічної гібридної архітектури реактивної системи керування польотом для БЛА як показує [18].

В [9], як і в [19], проблема посадки розглядається в його чутливості оцінки руху з відновленням руху камери за допомогою вимірювань нерухомих точок в середовищі зображення шляхом розробки алгоритму, специфічні для плоского випадку, назва якого "диференціал" версія. Лінійні і кутові швидкості камери відновлюються з використанням швидкості зображення нерухомих точок в площині, оскільки, коли всі особливі точки на зображенні лежать в одній площині особливий випадок виродження результатів, що робить "алгоритм 8-точок", погано обумовлений наданням бідними результатами оцінки, а отже обмеженням класичного випадку його алгоритму оцінки. Для мобільних роботів, диференціальна версія є більш відповідним для розробки контролера в якості оцінки швидкості.

[19], [9] виявили, що найбільш інтенсивно обчислювальні задачі як для дискретного, так і для диференціального випадку, є оцінка лінійної методом найменших квадратів з "плоскої матриці", яка містить всі параметри руху і структури, що підлягають корекції; а також плоска матриця є істотно диференціальна, яка містить всі параметри диференціального руху, необхідні для оцінки руху та стану. Вони обидва вимагають сингулярне розкладання (SVD) матриць, яке веде до зростання функції витрат алгоритму бачення. І виявили, що за допомогою бібліотеки MATHLIB C ++ в Matlab на 450MHz Pentium II під керування Linux, алгоритми бачення може виконати оцінку руху за допомогою 25 характерних точок при швидкості понад 150 Гц [19], [9].

Використовуючи критерії аналізу продуктивності, як (а) помилки оцінки залежать від різних рухів камери щодо площини спостереження, (б) оцінка ефективності в присутності шумів зображення при вимірах, і (в) для всіх симуляцій, порівнюючи результати з традиційний алгоритм 8-точок, вона була створена з [19] і [9] і змодельована, що (а) помилки в обертальних компонентах матриць не повинні залежати від глибини точок; (б) якість сигналу щодо шуму збільшилася, а продуктивність алгоритму знизилася; (в) для дискретних і диференціальних випадках планарний (по площині) алгоритм є краще, ніж алгоритм 8-точкової чутливості шуму; і (г) площинний алгоритм виконується краще, ніж 8-точкий алгоритм за винятком того, коли поступальна і оптична осі є паралельні.

Варто відзначити, що роботи [19] і [20] засновані на заздалегідь визначеному місці посадки. Це робить навігацію до місця посадки простіше через розташування позиції посадки по заздалегідь заданими характеристиками і безперервно оцінюючи його рух до цілі. Необхідність планування шляху і керування польотом таким чином усувається на відміну від посадки на рухому мішень або на невизначену місцевість.

* + - 1. **Слідкуючі системи технічного зору в контурі посадки безпілотних літальних апаратів**

Слідкуючий зір в контурі посадки БПЛА є тим підходом, який в повній мірі інтегрують систему комп’ютерного зору з або без інших інерційних приладів вимірювань в систему керування посадкою. Унікальні можливості БПЛА / вертольота зависати, вертикальний злет над землею, виконувати поздовжні і поперечні маневри польоту мають свою ціну: (а) вони споживають багато енергії; (б) вони є нестабільними і маневрування є складним без використання хороших контролерів; і (в) ємність корисного навантаження вкрай обмежена в результаті додаткових витрат.

Зір на основі самооцінки дозволяє замінити та відновити дані з інерційних датчиків дрейфових складових [17].

В [14], (12) представлено розробка і реалізація візуальних орієнтирів при посадці в кооперативному середовищі, де використовується швидкий, надійний і обчислювально легкий алгоритм, щоб приземлити вертоліт на мішень, що складається зі спеціальних маркерів. Камера утримується перпендикулярно поверхні землі в процесі. Аналіз зображень низького рівня був виконаний застосування 7х7 медіанного фільтру для видалення шуму. Порогами зображення потім вказували на підставі 8-точок, після чого були вилучені за допомогою перших інерційних інваріантних моментів третього порядку функції. Ці функції були використані для розпізнавання об'єктів і оцінки стану між автономного каліброваного зображення вертолітної камери. Зорова система поєднується з "низьким рівнем постурального контролю" для досягнення точного визначення цілі та її відстеження в той час, коли вертоліт оновлює свої цільові параметри посадки на основі бачення; він використовує контролер, заснований на оцінки стану, щоб слідувати до місця посадки - "колективний" контролер, який використовує різні елементи керування низького рівня в різних слабо пов'язаних розділах для досягнення спільної мети керування більш високого рівня. Підхід, який використовується для вилучення ознак в [14], має недолік, що полягає в зниженні впливу особливостей, які були вилучені, оскільки він використовує медіанний фільтр для видалення шуму, який є нелінійним фільтром, який видаляє значні особливості зображення.

Завдяки використанню підходу самооцінки, вертоліт був ініціалізований в зависанні в довільному місці, з якого він автоматично визначає вертолітний майданчик посадки, відстежуючи його і потім здійснювати посадку. Вертоліт, в ході льотних випробувань, визначив вертолітний майданчик і він здійснив посадку в межах 31см точності від позиції. Робота Монтгомері є унікальна тим, що вона виходить за рамки робіт, раніше виділених для демонстрації надійності алгоритму, щоб знайти вертолітний майданчик після того, як тимчасово втратити його з поля зору і здійснити посадку на рухому мішень тільки після зупинки.

Піддослідний є радіокерована модель вертольоту з газовим живленням, оснащена стеком PC-104 доповненої з датчиками. Boeing GMIGTS-II INS блок з трьома осями гіроскопа, інформацією про стан на комп'ютері в той час як Novatel RT-20 DGPS оцінює позиціонування з точністю до 20 см круговою ймовірністю. Ноутбук служив як відправник керуючих команд високого рівня і коригування диференціальної GPS на вертольоті. Зв'язок був реалізований ​​за допомогою бездротового Ethernet 1.8 ГГц і бездротове відео 2,4 ГГц, також автономний політ був досягнутий за допомогою контролера на основі стану ЛА.

Інші роботи, такі як [17] скористалися системою бачення не тільки при посадці, але в типовому підході одночасної локалізації і відображення (SLAM) в побудові карти підсистеми і підсистеми навігації.

Практичне значення мають проекти Хінце і О. Аміда [10], [17], які продемонстрували повну автономну посадку безпілотних літальних апаратів. В [10] БПЛА вилетів спочатку на віддалений майданчик посадки в середовищі моделювання під назвою Riptide - комп'ютер на основі моделювання навколишнього середовища, що працює в режимі реального часу, візуальний, повний діапазон режимів польоту, пілот або оператор-в-циклі моделювання легко доступні протягом усього циклу розробки. Спочатку дані надходили з GPS, а потім використовували алгоритми стереозору для створення 3D-інформацію про середовище, з якої була обрана точка посадки і перетворюється в 2D з використанням алгоритмів детермінації з безпечної зони посадки для пошуку даних 3D, які раніше згенерувалися. Алгоритми Монокулярної позиції відстежували 2D розташування пікселів для використання в якості фільтра Калмана, який оцінював швидкості і використовував їх в якості внесення до законів створення нормального шляху через шляхові точки до місця посадки. Робота Hintze не без обмежень. Його робота передбачає наступне: не-кооперативне місце посадки, що означає, коли транспортний засіб піднімається на велику висоту, стерео діапазонні алгоритми не дають точних оцінок для дрібних об'єктів на землі. Це є недоліком, оскільки об'єкти, які не захоплені з великих висот за допомогою стереосистеми бачення, можуть створити небезпеку при посадці; (Б) конструкції, такі як антени і електричні дроти важко відрізнити від фону - небезпека для транспортного засобу в режимі спуску, якщо моделі не чітко виділяються; і (в) вроджена проблема монокулярна оцінка становища, яка оновлює шаблон відповідності нової функції трекера після кожного кадру камери. Неправильний результат впливу до шаблону може статися через непередбачену обструкцію місця посадки.

При здійсненні такої конструкції, можна було б використовувати надлишкову систему трекерів таких, що у відмові одного трекера, інші можуть надати точну інформацію.

Аміди та ін. [17], [16] дають демонстрацію повної оцінки положення при посадці на основі зору з використанням платформи Berkely Yamaha R50 [23], а також дві камери Sony XC75, які використовуються візуальним одометром машини отримання зображень від двох незалежних A / D конвертерних модулів. A / D-модулі, які надають оцифровку зображення в режимі реального часу, реконфігуровані зображенням гасіння, і високошвидкісними портами зв'язку, які корисні для високошвидкісної цифрової обробки. У режимі реального часу модуль зображення конвольвер, інтегральна схема, GEC Плессі 16488 були використані для згортки зображення на частоті 10МГц в секунду для трафіку даних вхідного зображення поставляючи 640MOPS. Використовуючи достатні можливості в природних сценах, щоб зафіксуватися на будь-яких наземних цілях, одометр оцінює вертольотний бічний рух і висоту над мішенню.

Цей алгоритм відстеження реалізує два основні напрямки виконання. Основний напрямок оцінює стан вертольоту, а другий вимірює швидкість і створює нові потенційні шаблони для подальшого відстеження. У стерео NTSC камери синхронізовані через генератори синхронізації, що працює таким чином, щоб запропонувати таке ж поле зображення після кожного відкриття затвора камери. Це гарантувало нове поле зображення на кожні 1/60 секунди. Підтримуючи блокування двох сегментів зображення або шаблонів і активне відстеження їх через високошвидкісну кореляцію, одометр був в стані відрізнити обертальний/поступальний рух від сприйнятого переміщення зображення за допомогою мічення зображень зі стану вертольоту. Тріангуляції зміщення зображення після цього використовується для виявлення нового положення вертольоту.

Результати були наступні: точність бічної і поздовжньої оцінки положення до 1,5 см, не дивлячись на частоту 1-3Hz, 5-10 ° амплітуди коливань. Була помітна помилка в поздовжньому напрямку 50-60% через більш низьку роздільну здатність зображення вздовж напрямку, яке мало вплив на шаблон кореляції точності позиціювання.

* + 1. **Методи виявлення кутів в алгоритмах посадки БПЛА**

Серед різних існуючих методів сегментації виявлення кутів [21], [22], [23], [24], [25] є дуже корисною для вилучення ознак з цілі посадки, оскільки вони несуть корисну інформацію про межі кривих і хороші в унікальності ідентифікації і відносної стабільності в зображеннях з невеликою дисперсією. Харріс [26], Розенфельд [24], і Бреді [27] запропонували методи виявлення кутів, але їх підхід з одним масштабуванням і працює добре, якщо функцій не так багато. Це є істотним недоліком, оскільки конструкція посадкової мішені обрана такою, щоб фон можна було б відрізнити від самої мішені.

В роботах Мокхтаріан [21], [22], [23] були знайдені надійні адаптації до задачі посадки цільової сегментації. В [21] представлено мультимасштабування, основна форма кривизни якої визначається методом уявлення для плоских кривих. Описуючи криві з різним ступенем абстракції на основі інваріантних ознак, представлені результати, які показують еволюцію довжини дуги, не змінюючи меж кута. Але робота Мокхтаріан тільки використовує багатомасштабний алгоритм для локалізації при виборі одномасштабних методів для виявлення кутових позицій. В результаті справжні кути втрачаються, коли суттєво зростає відхилення і навпаки, визначаються неправдиві кути при низькому відхилені.

В [22] поліпшена робота Мокхтаріан: замість обчислення кривизни окремих контурів, при низькому рівні з перетворенням по критерію глобального порога, відбувається порівняння з адаптивним локальним порогом замість глобального перед визначенням ймовірних кутів локальним максимом абсолютної функції кривизни. [29]. Ймовірні кути проходять перевірку, щоб усунути помилкові кути, джерелом яких є динаміка польоту. Результати показують, що була виявлена найбільша кількість дійсних кутів, помилкові кути були значно мінімізовані і результуючі кути уважно стежили за опорними кутами з мінімальним виявленням неправдивих кутів.

* 1. **Розгляд найближчих аналогів проекту**
     1. **Автоматична посадка на рухому мішень для безпілотних вертольотів на основі технічного зору [30]**

Основний напрямок у дисертації зроблено на пропозицію архітектурного дослідження для посадки на рухому мішень на основі технічного зору, особливо для безпілотних вертольотів. Вона охоплює закон наведення, обробки зображення, оцінки стану, моделювання та проведення льотних випробувань.

Закон наведення для посадки на рухому ціль можна розділити на два етапи. Перший етап «наближення гелікоптера до цілі». Були запропоновані дві концепції на основі цільових алгоритмів спостереження. Один алгоритм був на основі морфології кольору і побітові оператори, щоб підвищити швидкість обробку зображення в режимі реального часу на платі. Технічний зір на основі алгоритмів відстеження дають цільовий стан інформації на піксельній площині камери або у вигляді вектору у кадрі камери, яка може бути перетворена в інформацію. Це дозволило реальної реалізувати запропоновані алгоритми. Було виконано реальні льотні випробування для перевірки.

Другий етап «посадка», коли алгоритми для першої фази не працюють із-за короткої відстані між БПЛА і цілі. Замість цього запропоновано на основі прийняття маркеру створюють алгоритм оцінки з відомою орієнтації плоских точок.

Подовжнє і поперечне керування були представлені з ідеєю, зберігаючи відносне положення постійним, створити додаткову команду підходу до БПЛА на ціль з використанням модифікованої лінії візуалізації вказівок. Команда зміни висоти була введена окремо, де БПЛА перемикається між фазами наближення. Всі роботи були інтегровані і перевірені за допомогою моделювання.

Недоліки даної роботи є використання алгоритму ідентифікації цілі за його кольором – даний метод є ненадійний, особливо в польових випробувань, оскільки дуже суттєво залежить від умов навколишнього середовища (яскравість, освітленість).

* + 1. **Автономна навігація вертольотного безпілотного літального апарату з використанням машинного зору [31]**

Проектування та реалізація алгоритмів комп'ютерного зору для автономної посадки гвинтокрилого безпілотного літального апарату.

Процедура посадки складається з двох етапів. Перший БПЛА переходить з початкового положення в заздалегідь визначене місце вертолітного майданчика, парирує над ним, і витягує кутові функції за допомогою кривизни масштабного алгоритму простору кута. Пізніше на детекторі налагоджується зображення для стандартного калібрувати вертолітного майданчика за допомогою алгоритму машинного навчання, обмеженою Машиною Больцмана. Другий етап передбачає високий рівень оцінки стану руху і стану літального апарата щодо приземлення.

Особливості зашумлених зображень в порівнянні з еталонним показали 100% кореляцію перекладу осі і 0.02º помилки обертання. Обмежена машина Больцмана для моделі навчання з різних тестових зображень показала 36% помилок класифікації, що є суттєвим недоліком в розпізнаванні місця посадки.

* 1. **Розв’язані питання в літературних джерелах**

Існують наступні алгоритми ідентифікації місця посадки: 1) рухомий майданчик з індивідуальним своїм кольором, який розпізнається, виділяється і його координати передаються в систему керування БПЛА типу квадрокоптер; 2)знаходження кутів на зображенні сцени, з яких отримувати кути інтересу для визначення відповідного майданчику посадки. Проводиться порогове перетворення в бінарну картинку для подальшого визначення цілі посадки, перед цим медіанним фільтром прибрати шуми зображення; 3) визначення контурів майданчика, використовуючи схожу модель обробки зображення, що описано в попередньому методі; 4) вилучення особливих ознак цілі посадки з кадра камери.

Також існують методи підбору технічного забезпечення СТЗ: підрахунки необхідних технічних вимог потужності мікропроцесорів обробки зображень та параметрів камери для знаходження місця посадки виходячи з розмірів та особливостей самого наземного символу.

Описані методи/підходи/етапи посадки БПЛА на основі СТЗ як інтегрованої в систему керування, так і окремо, похибка яких не більше 31см по повздовжньому русі та 6º бічному відповідно. Розв’язані задачі посадки з високих висот, але при малих висотах, коли камера вже не може захватити місце посадки, задача посадки розв’язується лише в деяких ситуаціях.

Розв’язані проблеми зв’язку та організації обміну даними БПЛА зі землею. Описані методи підвищення надійності вдалого виконання місії БПЛА при використанні СТЗ в контурі керування.

**ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ**

З розділів 1.1 і 1.2 видно, що використання систем технічного зору в заході на посадку безпілотного літального апарату усувають помилки, які пов'язані з інерційними інструментами вимірювання. Більшість дослідників були зосереджені на посадці БЛА на заздалегідь визначені цілі та з використанням інших способів цільової посадки, місцезнаходження якої апріорно не відоме. Очевидно, що важко створити візуальний алгоритм, який може знайти безпечне місце посадки на основі тільки однієї системи технічного зору.

На основі різних підходів, результати оцінки для положення і орієнтації БЛА / вертольота щодо цілі посадки стає більш точним за допомогою класичного диференціального алгоритму оцінки стану для мобільного робота. Наприклад, коли Шакернія і ін. [19] продемонстрували свою роботу в [20], вони на основі технічного зору оцінили точність посадки до 5 см і 5º по поступальній та обертальній осях відповідно. Це тому, що їх підхід оцінювання розроблено в повному сенсі: динамічна модель БЛА з баченням на основі позиційного вимірювання - самокорегуюча на основі візуальної фіксації на особливості місцевості. Результати руху були використані при конструюванні контролера. В. [14], [26] використовується контролер на основі стану руху, який розділений загальним контролером в набір різних допоміжних контролерів після формування інваріантного моменту функції сегментації. Їх результати були точні лише на 31см позиційно і 6º обертання відповідно. Коли інваріантні моменти використовуються для вилучення характерних точок у цілі посадки, ознаки мають тенденцію бути більш відмітним і характерним, як показано в Монтгомері і ін. [17, 25], де вони оцінили точність між «офлайновими» відкаліброваними інваріантними моментами та моментами, які генеруються в польоті - в межах величини допуску ± 10%.

За результатами дослідження даного розділу можемо зробити наступні висновки про нерозв’язні задачи в даній темі.

Суттєвою проблемою використання СТЗ в керуванні БПЛА є його синхронізація органів керування (гвинти-двигуни) з комп’ютерним зором.

Наступна нерозв’язана задача залишається в оптимізації зображення місцевості, оскільки на СТЗ дуже суттєво впливає зміна стану навколишнього середовища (освітленість, яскравість, інтенсивність). Алгоритми ідентифікації об’єктів є дуже чутливими до цих параметрів, а при неоптимальній обробці зображення можуть виникати шуми на зображенні. Також проблемою є підбір необхідних фільтрів шумів, оскільки неправильна фільтрація може призвести до втрати важливої інформації з зображення місцевості, що в свою чергу може призвести до нездатності знаходження цікавлячих об’єктів.

Проблема надійності даних систем: використання одного методу ідентифікації цілі є не завжди надійним, тому краще комбінувати різні методи, особливо різні за їх природою розпізнавання. Більшість проектів використовують тільки один метод розпізнавання.

Повністю невирішена задача посадки БПЛА на малій висоті на основі СТЗ без використання інерційних датчиків/висотомірів, що суттєво зменшує точність посадки в існуючих алгоритмах керування БПЛА при його знижені.

Також проблематикою залишається вибір відповідного алгоритму ідентифікації цілі та саму ціль (його параметри та особливості), виходячи з поставлених задач, оскільки вони не є універсальні для кожного БПЛА та характеристик місцевості.

В даній магістерській роботі запропоновані методи та алгоритми вирішення нерозв’язних задачи вище, розробивши систему автоматичного керування БПЛА за допомогою СТЗ для вирішення задачі його автоматичної посадки на спеціально підготовлену площадку в реальному масштабі часу з заданою точністю.

Для вирішення даних проблем в даній роботі розроблено алгоритм автоматичної посадки БПЛА типу квадрокоптер на основі СТЗ на спеціально визначене місце, яке має свою особливу символіку (символ і певні позначки).

Пропонується комбінований спосіб (який не залежить від глибини зображення для уникнення зайвих шумів) ідентифікації місця посадки, оснований на декільком методів розпізнавання як символу, так і певних особливостей/ознак майданчику посадки, для підвищення надійності даного алгоритму, а також введення в систему керування БПЛА типу квадрокоптер закону виконання стабілізації в непередбачуваних ситуаціях (при втраті зображення чи цілі).

Підбираються та вводяться нові методи фільтрації для оптимізації та покращення зображення, максимальне видалення шумів. Комбінація даних фільтрів з методами розпізнавання місця посадки є особливим в даній роботі, оскільки йде підбір найбільш оптимального знаходження місця посадки в умовах всепогодності навколишнього середовища при умові, що є освітленість.

Вирішується проблема посадки на малих висотах за методом «target in target» (новий метод: розміщення об’єкту розпізнавання в самій символіці місця посадки, який можливо побачити тільки з малих відстаней, а також новий багатомасштабований алгоритм розпізнавання символу). Очікується точність посадки між центром майданчика та вертикальної осі БПЛА.

1. **АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМИ ТЕХНІЧНОГО ЗОРУ**

Типове рішення системи машинного зору включає в себе декілька наступних компонентів [32]:

1. оптичні пристрої (або оптична система в цілому) – одна або кілька цифрових або аналогових камер (чорно-білі або кольорові) з відповідною оптикою для отримання зображень;
2. програмне забезпечення виготовлення зображень для обробки. Для [аналогових камер](https://uk.wikipedia.org/w/index.php?title=%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D0%B0_%D0%BA%D0%B0%D0%BC%D0%B5%D1%80%D0%B0&action=edit&redlink=1) це є [оцифрування](https://uk.wikipedia.org/wiki/%D0%9E%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) зображень;
3. процесор (сучасний [ПК](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) з багатоядерним процесором або вбудований процесор, наприклад — [ЦСП](https://uk.wikipedia.org/w/index.php?title=%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80&action=edit&redlink=1));
4. програмне забезпечення машинного зору, яке надає інструменти для розробки окремих застосувань програмного забезпечення;
5. устаткування введення/виведення або канали зв'язку звіту про отримані результатів.

СТЗ працює за наступним принципом. Зображення з камери потрапляє в захоплювач кадрів(Image Capture) або в пам'ять комп'ютера у системах, де захоплювач кадрів не використовується. Захоплювач кадрів - це пристрій оцифровки (як частина камери або у вигляді окремої плати в комп'ютері), яка перетворює вихідні дані з камери в цифровий формат. Як правило, це двовимірний масив чисел, що відповідають рівню інтенсивності світла певної точки в області зору, званих пікселями. Цей масив розміщує зображення в пам'яті комп'ютера так, щоб воно могло бути оброблено за допомогою програмного забезпечення для машинного (технічного) зору.

В загальному випадку на борту ЛА використовують оптичний датчик (камера) з електричною схемою, яка організовує первинну обробку зображення та передачу цих даних на ЕОМ, яка перетворює в потрібний нам вид сигнал від камери і в якому знаходяться відповідні алгоритми обробки зображення розпізнавання цікавлячих нас об’єктів, а далі, або цей контролер (ЕОМ) приймає рішення сам на базі вкладених в нього алгоритмів і формує відповідні сигнали керування ЛА, або ж він передає інформацію із зображення пілоту літака(або спостерігачу), а той уже приймає рішення про подальші дії. Прикладом є направлення БПЛА на посадку в певну точку на поверхні Землі, що і є задачею вирішення даної роботи [33].

* 1. **Оптико-електронні системи в СТЗ**

Оптико-електронна система включає оптичні пристрої(камера з об’єктивом) і електронну систему оцифровки зображення і первинної обробки її.

Оптичні системи в безпілотної техніці застосовуються для:

- огляду місцевості і спостереження за обстановкою на ній;

- отримання детального зображення ділянок місцевості та об'єктів, що знаходяться на ній;

- розкриття візуально невидимих об’єктів, що перебувають на місцевості.

Технічно системи реалізуються у вигляді системи оптичних пристроїв, які розміщуються або безпосередньо в корпусі БПЛА і жорстко пов'язаних з його конструкцією, або за допомогою гіростабілізованих платформ, що забезпечують поворот оптичних осей пристроїв в будь-який бік із заданою швидкістю.

Термін «жорсткий зв'язок» є умовним. Майже всі оптичні пристрої, установлювані в конструкцію БПЛА, мають можливість деякого кутового переміщення для компенсації кутового положення цього БПЛА відносно поверхні Землі і кутових коливань БПЛА відносно осей нормальної СК [34].

Оптичні пристрої з жорстким зв’язком дозволяються вирішувати велику кількість практичних завдань, однак для отримання якісного результату необхідно виконувати вимоги до стабілізації положення БПЛА в повітрі в момент знімання. На жаль, принцип побудови цих пристроїв зазвичай не дозволяє реалізувати режими супроводження цілей, а в разі спроби реалізації цього режиму з’являється велика кількість обмежень і складностей.

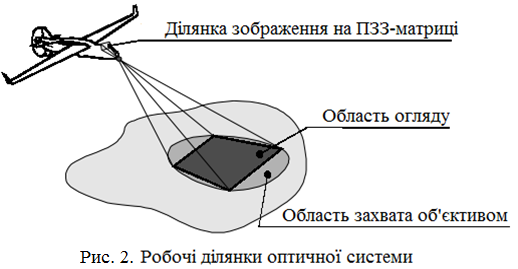
Більш гнучкими виявляються оптичні пристрої, розміщені на гіростабілізованих платформах.

* + 1. **Роздільна здатність оптико-електронних систем**

Під роздільною здатністю цифрової оптичної системи слід розуміти здатність оптичної системи формувати на пристрої виводу (наприклад, на екрані) зображення предметів, віддалених на задану відстань від об'єктива оптичної системи із заданим рівнем деталізації. Основним чинником, що визначає роздільну здатність оптико-електронної системи є ПЗЗ-матриця, на яку після об'єктива проеціюється одержуване зображення.

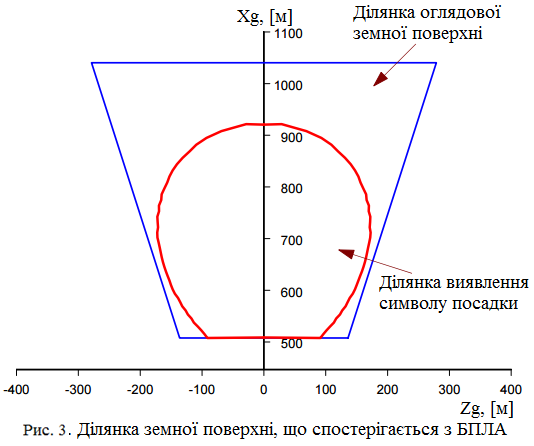
Джерелом видової інформації є земна поверхня. Кути поля зору об'єктиву визначають обсяг інформації, що знімається з земної, і впливають на роздільну здатність пристрою. Варто також враховувати, що видова інформація, що виводиться на екран, відрізняється від фізичного змісту ділянки місцевості як за передаванням кольору [35], так і щодо відповідності зображення на екрані реальним лінійним розмірами й обрисами об'єктів на земній поверхні. На рис.1 показані області перегляду, формовані об'єктивом оптичної системи. Розміри області перегляду залежать від висоти польоту ЛА, кутів поля зору об'єктиву і кута нахилу оптичної осі об'єктива.

На рис.2 показаний варіант реалізації оптико-електронного пристрою. Видно, що форма фізично видимої області ділянки деформується в фігуру близьку до трапеції.



При розрахунках, в першому наближенні, можна використовувати допущення, що через малість співвідношення :

де – довжина, що дорівнює найбільшому лінійному розміру області перегляду, – середній радіус Землі, спотвореннями через кривизни земної поверхні можна знехтувати і представляти її площиною, що збігається з деякою серединною поверхнею.



* Висота польоту – 650 м;
* Швидкість польоту – 540 км/год;
* Кут нахилу оптичної осі – ;
* Кут поля зору - ;
* Формат зображення -

Однак реальний профіль земної поверхні далекий від площини і це сильно позначається на ефективності використання оптико-електронних пристроїв. Враховуючи, що бортові оптико-електронні пристрої застосовуються не для споглядання навколишньої місцевості, а для виконання конкретних функцій, то доцільно розглядати ступінь придатності оптичної системи по можливості виявлення типового об'єкта в типових умовах застосування БПЛА. Наприклад, на рис.2 показана область, в якій можна виявити символ посадки (літера «Н»), якщо він там є, наприклад, при заході на посадку даного БПЛА [36]. Межа цієї області є сукупністю координат, в яких даний символ видно як точку (піксель).

Одним з основних параметрів, що дозволяють визначити необхідну висоту польоту в залежності від можливостей встановленої оптичної системи, є коефіцієнт заповнення (рис.3):

де – площа області, в якій видно ціль (об'єкт), – площа області перегляду.



Величина, обернена коефіцієнту заповнення показує, у скільки разів необхідно збільшити площу області для огляду, щоб шуканий об'єкт (ціль) можна було побачити на екрані оптичної системи. Розмірність ПЗЗ-матриці значно впливає на роздільну здатність оптичного пристрою (системи), оскільки розмірність ПЗЗ-матриці визначає рівень дискретизації зображення, яке надходить з об'єктива. Чим вища розмірність матриці, тим вище рівень дискретизації і тим більше деталізується зображення на екрані системи. Отже, щоб отримати роздільну здатність цифрової оптичної системи, близьку до оптичної роздільної здатності об'єктива треба або мати ПЗЗ–матрицю дуже великої розмірності або зменшувати кути поля зору [32].

Роздільна здатність цифрової оптичної системи (об'єктив -ПЗЗ-матриця – пристрій виведення зображення (екран) у загальному випадку визначається як залежність середньої по екранному полю величини локальної роздільної здатності від групи параметрів:

де - похила дальність до точки на земній поверхні, , , - відповідно кутовий зсув розглянутої точки від оптичної осі вправо-вліво, вгору-вниз і кут нахилу оптичної осі в нормальній СК, – розмірність ПЗЗ–матриці. Очевидно, що розмірності ПЗЗ - матриці і матриці екрана повинні бути близькі, а точніше - розмірність матриці екрана повинна бути не нижче, ніж у ПЗЗ-матриці.

* + 1. **Методика вибору характеристик оптичної системи**

Вибір характеристик оптичної системи для БПЛА залежно від висоти польоту можна здійснити таким чином [37], наприклад: камера використовується з об’єктивом, який дає десятикратне оптичне збільшення K = 10; розмір кадру А х В (4000 х 3000) пікселів (12 мегапікселів); матриця камери має розмір a x b (18 х 13,5) мм.

Для визначення розміру пікселя ( R ) необхідно поділити розмір матриці в міліметрах на розмір кадру в пікселях, причо му береться менший розмір (у даному випадку 3000 пікселів).

R = b / B = 13,5 / 3000 = 0,0045 (мм). (4)

Фокусна відстань ( F ) визначається, виходячи з оптичного збільшення:

F = b / K = 13,5×10 = 135 (мм); (5)

в пікселях

Fпікс = ( b / R ) K = ( 13,5 / 0,0045) = 30000 (пікс.). (6)

Для цифрової зйомки використовується поняття «розмір пікселя на місцевості» (у англомовній літературі GSD – ground sample distance). GSD визначає точність результатів подальшої обробки, при зйомці з висоти:

GSD = H / Fпікс (7)

де H – висота, см; F – фокусна висота, пікс.

Для зйомки з висоти 500 метрів над землею (50000 см) величина GSD буде дорівнювати:

GSD = 50000 / 30000 = 1,67 ().

Якщо відома GSD , то можна визначити геометричні розміри поверхні, яка попадає у кадр:

– довжина L = A GSD = 4000 × 1,67 = 6680 (см); (8)

– ширина M = B GSD = 3000×1,67 = 5000 (см). (9)

При виводі зображення на монітор або для друку фотографій у пікселів з’являється фізичний розмір. Чим більше пікселів на дюйм (англ. – pixels per inch-ppi), тим менш помітними будуть окремі пікселі й тим більш реалістичним буде виглядати зображення. Для отримання якісного зображення в одному дюймі повинно міститися 300 точок (пікселів).

Оригінальний розмір зображення:

– довжина зображення L1 = 4000 / 300 = 13,3 (дюйм) = 33,87(см);

– ширина зображення M1 = 3000 / 300 = 10 (дюйм) = 25,4(см).

Наприклад, задача моніторингу полягає у визначенні людей в зоні спостереження (наприклад, 2 × 0,6 м), тоді:

Х = 2 × 25,4 / 50 = 1,02 (см);

Y = 0,6 × 33,87 / 66,8 = 0,3 (см).

Дана методика дуже проста та дозволяє порівнювати різні характеристики оптичної системи. Аналізуючи отримані результати, можна зробити такі висновки: використовуючи оптику, яка має більше оптичне збільшення, отримаємо на зображенні більший розмір необхідного нам об’єкта (людини); при більшому оптичному збільшені розмір земної поверхні, що потрапила в кадр, зменшується; використовуючи камеру з більшою кількістю пікселів, збільшуємо розмір зображення на екрані.

* 1. **Обчислювальні системи безпілотних літальних апаратів як частина системи технічного зору**
     1. **Архітектура бортових обчислювальних систем**

Основними особливостями побудови сучасних обчислювальних систем БПЛА є [34]:

* компактність розташування компонентів КБО;
* наявність потужних мікропроцесорів ЦОС, що мають вбудовані засоби створення багатопроцесорних обчислювальних систем.

Під час формування систем авіоніки конкретного БПЛА виникає потреба у виборі мікроконтролера, що задовольняє вимоги до продуктивності , надійності, умов застосування і т. ін.

Основні вимоги до мікроконтролерів такі:

* тип виконуваних операцій (числова або бітові);
* кількість маніпуляцій для оброблення даних;
* робота системи (щодо переривань, готовності або команд людини);
* кількість пристроїв/бітів введення-виведення (І/О), якими необхідно керувати;
* тип пристроїв серед багатьох можливих типів І/О пристроїв, які повинні контролюватися/керуватися;
* напруги живлення, необхідні для системи;
* діапазон коливання напруги живлення;
* робочий струм;
* обмеження щодо розмірів, маси, естетичних параметрів, таких як форма і колір;
* специфічні вимоги до умов навколишнього середовища, таких як військові умови, температура, вологість, атмосфера (вибухонебезпечна, корозійна і т. ін.), тиск і висота.

Оскільки основна частина периферійних пристроїв керується з бортового обчислювача, то найбільш трудомістким етапом проектування системи керування БПЛА є розроблення програмного забезпечення.

У процесі вибору апаратних засобів для створення нового БПЛА необхідно розглядати не тільки технічні засоби мікроконтролера, але також приділяти увагу засобам підтримки – як апаратним (стартові набори, програматори, схемні емулятори), так і програмним (мови низького й високого рівня, емулятори). Для розрахунків варто брати не тільки зручність роботи й функціональні можливості конкретного пакета програми, але і його вартість.

* + 1. **Операційні системи бортової авіоніки безпілотних літальних апаратів**

Нині інтенсивно ведуться роботи зі створення ІМА. Основними відмітними концептуальними особливостями архітектури БОС на ІМА варто вважати:

* реалізованість, відкритість і адаптованість архітектури та масштабованість обчислювальних засобів;
* спільність використання на борту різних БПЛА апаратних і програмних засобів;
* незалежність програм від використовуваних апаратних платформ;
* прийняту вартість елементної бази БОС;
* орієнтацію на широке використання комерційних технологій і компонентів;
* уніфікованість мережі передавання даних;
* стандартизацію конструкції;
* високий рівень надійності, ремонтопридатності, технічного обслуговування, ефективні засоби ізоляції помилок.

Ядро авіоніки повинно бути апаратно-незалежним, тобто виконувати свої завдання незалежно від типу обчислювального середовища (мікроконтролерів, датчиків, сервоприводів) у конкретній реалізації автопілота.

Центром системи є ядро авіоніки – програмний код, що містить алгоритми керування БПЛА. Ядро являє собою частину БОС, у якій забезпечується вирішення завдань, максимально незалежне від конкретного складу бортового обладнання. Фактично ядро і є САК БПЛА. Для роботи САК необхідні вихідні дані датчиків і виходи до сервоприводів РМ керування. Посередниками між ядром авіоніки й апаратними засобами конкретної реалізації автопілота є процесори оброблення даних (у цьому випадку – процесори вимірювання й керування).

Саме розробка програмного забезпечення є найбільш громіздкою роботою у створені СТЗ, тобто розробка СКЗ, в якій закладено необхідні алгоритми трансформацій зображення та пошуку цікавлячих об’єктів, а також оцінка даної місцевості і видача необхідних сигналів керування на органи керування.

**ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ**

В даному розділі було проаналізовано типові рішеня СТЗ, що складаються з оптичних пристроїв, ПО, процесора обробки інформації та каналів зв‘язку. Розкрили зміст застосування оптичних систем: огляду місцевості і спостереження за обстановкою на ній, отримання детального зображення ділянок місцевості та об'єктів, що знаходяться на ній та розкриття візуально невидимих об’єктів, що перебувають на місцевості.

Проаналізували вплив роздільної здатності оптико-електронних систем на можливості та функціонал встановленої оптичної системи через коефіцієнт заповнення та висота БПЛА. На основі даної інформації було розроблена методика вибору характеристик оптичної системи, маючи всі необхадні параметри оптичного датчика.

Також, розглянули основні особливості побудови сучасних обчислювальних систем БПЛА та вимоги до мікроконтролерів щодо обробки зображення в реальному масштабі часу. У процесі вибору апаратних засобів для створення нового БПЛА необхідно розглядати не тільки технічні засоби мікроконтролера, але також приділяти увагу засобам підтримки – як апаратним (стартові набори, програматори, схемні емулятори), так і програмним (мови низького й високого рівня, емулятори).

1. **ВИБІР СКЛАДОВИХ СИСТЕМИ ТЕХНІЧНОГО ЗОРУ**
   1. **Вибір оптичної системи**

Потрібно вибрати оптичний датчик, який зможе «побачити» наш символ «Н» для подальшого оброблення цієї картинки і розпізнавання цього символу літальним апаратом.

Для більш-меншого розпізнавання символу «Н» на матриці камери потрібно, щоб квадрат посадки займав мінімум 20х20 пікселів; рекомендується для таких проектів брати розмір символів із запасом, тому було обрано 50х50 пікселів мінімального розміру символу на зображені на висоті 15 метрів для даної системи.

Щоб на висоті *H* = 15 м квадрат 0.7 × 0.7 м мав розмір на картинці 50х50 пікселів, потрібно забезпечити:

Маємо співвідношення:

де *H* – висота, *F* – фокусна відстань камери, *B* – розмір кадру в пікселях для меншої сторони, *b* – розмір матриці камери.

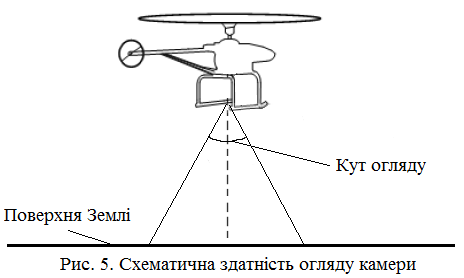
*K* – оптичне збільшення об’єктива камери.

Зауважимо, що основна похибка вимірювання оптичного датчика може бути в 1 - 2 пікселя, а вже яка дана похибка на місцевості залежить від висоти.

При виборі камери найбільш важливі параметри саме є фокусна відстань *F*, кількість пікселів *B* і розмір матриці *b* ( ці параметри присутні майже у всіх магазинах для купівлі). Тому для H = 15 м і GSD = 1.4 см:

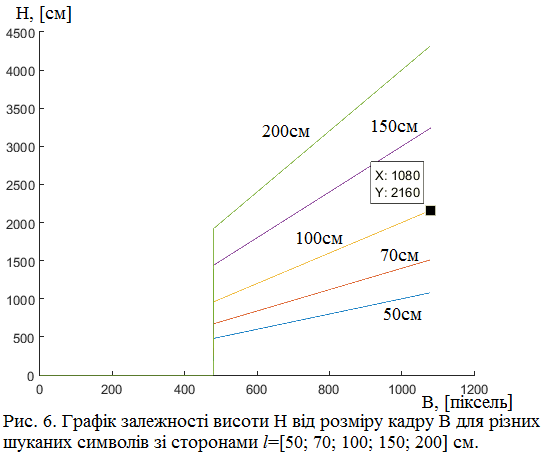
Наприклад, для камери, фізичний розмір матриці якої по діагоналі 1/3", *a* × *b* = 4.8 × 3.6 мм, *А* × *B* = 640 х 480 пікселів, необхідна фокусна відстань об’єктива:

Але в прикладі вище закладено *К* = 2.2, а це означає, що кут огляду місцевості з БПЛА зменшується, тобто:



при збільшені оптичної здатності камери, зменшується кут огляду, а, отже, зменшується площа захвату ділянки поверхні Землі.

Тому для даної задачі найбільш ефективним буде *К* = 1, тобто *F*=*b*.



Тоді, співвідношення (11) разом із формулою (10) набуде кінцевого вигляду:

При збільшені К, допустима висота знаходження символу збільшується на таку ж величину.

Максимальна сторона зони огляду розраховується:

де *R* - розміру пікселя, мм;

- фокусна відстань в пікселях.

З формули (2):

і із співвідношення (1) маємо:

Отримали схоже, що й з максимально допустимою висотою (формула 15).

Кут огляду ділянки при *К*=1 буде 53°.

Отже, виходячи з розрахунків вище і ТЗ, маємо наступні характеристики камери (разом з об’єктивом):

* роздільну здатність не менше 640 × 480 пікселів;
* матриця ПЗЗ не менше 1/3” дюйма;
* фокусна відстань: від 2.9 до 8 мм; бажано автофокус;
* габаритні розміри: 60×50×40 мм.

Для збільшення можливою висоти розпізнавання символу при наявності даної камери можна збільшувати розмір відповідного символу. Так, для розмірів символу 200 × 200 см при оптичному збільшені *К* = 1 (максимальний кут огляду) максимальна висота буде 20 м.

Для перевірки алгоритмів СТЗ і САК була обрана камера Super WDR 800TVL Effio-V SONY CCD 28X28mm Camera Starlight 0.0003Lux:



Рис. 7. Обрана камери для тестування

Особливості камери:

* 1 / 3''SONY 960H WDR CCD (662/663);
* DSP: CXD4141GG;
* 3.6 мм об'єктивом, Додатковий 2,8 / 6/8 мм;
* OSD меню налаштувань лінії;
* Супер Широкий динамічний діапазон (WDR);
* Придушення шуму 2D і 3D (3D-DNR);
* Адаптивне відтворення тону (ATR-EX);
* Висока компенсація стрічного засвічення (HLC);
* Автоматичний баланс білого (White Bal);
* Автоматичне регулювання посилення (AGC);
* Білі цяточки пікселів Fix (W.PIX МАСКА);
* Довга витримка (Sense-вгору);
* 255X Цифровий зум і видошукач;
* Функція Антивуалент;
* Перевер.ізобр Функція (Flip);
* Цифрова стабілізація зображення Функція (ДІС);
* Функція Privacy Mask (Маска конфіденційності);
* Мови: китайська, англійська, японська, французька, португальська, німецька, російська, іспанська, і т.д .;
* DC 12V Потужність;
* вихід BNC відео.

Виходячи з різних завдань, які необхідно вирішити з використанням СТЗ, можна виявити необхідні характеристики камери з методу, яка представлена вище в цій роботі.

ОД для безпосереднього використання на БПЛА буде обиратися в майбутньому в наступній роботі, виходячи з даного методу вибору камери.

* 1. **Вибір електронної обчислювальної машини**

Оскільки дана система технічного зору повинна працювати в реальному масштабі часу, то потрібно забезпечити повністю мінімальні потреби оптичного датчика щодо мікроконтролера.

Для того, щоб МК мав можливість працювати з камерою в реальному масштабі часу (перегляд відео), потрібно забезпечити такі вимоги МК [41]:

1. Процесор (CPU):
   1. Рекомендований – Intel Pentium 4 2.0 ГГц чи аналогічний (ARM Cortex A15+);
   2. Мінімальний - Intel Pentium 3 800 МГц чи аналогічний (ARM Cortex A7+);
2. Пам’ять (RАM):
   1. Рекомендована – 2 Гб;
   2. Мінімальна – 256 Мб;
3. AGP відео карта з 64 Мб відео пам’яттю (128 МБ рекомендовано);
4. Bus Configuration: USB 2.0 чи 64-бітний PCI для IEEE-1394 карти (мінімальний 32-бітний);
5. Microsoft Visual C++ 6.0 або аналог (для компіляції і запуску коду програм, пов’язані з використанням камери);
6. По можливості графічний прискорювач GPU + DSP процесор.

Чим потужніше МК, тим більш надійно і значно краще буде працювати система в цілому, оскільки більша здатність здійснення операцій в один момент часу дає можливість швидше отримувати кадри с камери, що підвищує ФПС, і слідом і час затримки корекції САК.

* + 1. **Вибір електронної обчислювальної машини для БПЛА**

З огляду на забезпечення надійності системи для БПЛА було обрано наступний МК: МОДУЛЬ NVidia JETSON TK1 [42].

NVIDIA JETSON - провідна у світі платформа візуальних обчислень, що представляє можливості паралельної обробки даних з прискоренням на GPU для вбудованих систем. Високопродуктивні обчислення з низьким рівнем енергоспоживання в області глибокого навчання і комп'ютерного зору роблять JETSON ідеальним рішенням для розробки додатків і систем, які потребують великого обсягу обчислень. До них відноситься створення таких проектів, як:

* Дрони;
* Автономні роботизовані системи;
* Мобільне обладнання для медичної візуалізації.

Платформа для розробок NVIDIA JETSON TK1 забезпечує все необхідне, щоб розкрити потужність GPU для розробки додатків для вбудованих систем.

Вона створена на базі революційної система-на чіпі NVIDIA Tegra® K1 і використовує той же обчислювальний ядро ​​з архітектурою NVIDIA Kepler, яке лежить в основі суперкомп'ютерів по всьому світу. Це забезпечує повнофункціональну платформу з підтримкою NVIDIA CUDA для швидкої розробки та розгортання систем, що вимагають великої обчислювальної потужності, в області машинного зору, робототехніки, медицини та багатьох інших галузях.

NVIDIA надає всі інструментальні засоби для розробки вбудованого ПО, включаючи CUDA, OpenGL 4.4 і прискорену бібліотеку OpenCV. Завдяки повному набору засобів для розробки і профілювання, а також вбудованої підтримки камер і інших периферійних пристроїв, компанія NVIDIA пропонує ідеальне рішення, яке допоможе сформувати майбутнє вбудованих систем.

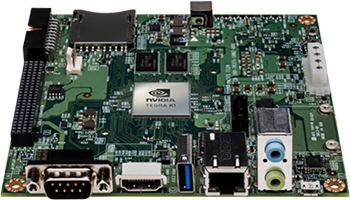


Рис. 8. NVidia JETSON TK1

Платформа для розробок JETSON TK1 включає:

* Систему-на-чіпі Tegra K1
* NVIDIA Kepler GPU c 192 ядрами CUDA
* Чотирьохядерний процесор ARM® Cortex ™ -A15 з архітектурою NVIDIA 4-Plus-1
* 2 ГБ пам'яті х16, 64-бітна шина
* 16 ГБ флеш-пам'яті, що відповідає специфікації eMMC 4.51
* 1 слот mini-PCIE половинного розміру
* 1 повнорозмірний слот SD / MMC
* 1 повнорозмірний слот HDMI
* 1 роз'єм USB 2.0 типу Micro-AB
* 1 роз'єм USB 3.0 типу А
* 1 порт послідовного інтерфейсу RS232
* 1 аудіо кодек ALC5639 Realtek з входом для мікрофона і лінійним виходом
* 1 мережевий адаптер RTL8111GS Realtek GigE LAN
* 1 порт SATA
* 4 МБ завантажувального флеш-пам'яті з інтерфейсом SPI
* Доступ до сигналів за допомогою порту розширення:
  + DP / LVDS
  + Touch SPI 1x4 + 1x1 CSI-2
  + GPIOs
  + UART
  + HSIC
  + I2c
    1. **Вибір електронної обчислювальної машини для лабораторного стенду**

Для тестування розроблених алгоритмів та їх налагодження достатній МК, який забезпечить мінімальні необхідні вимоги, щодо роботи СТЗ.

Огляду на це, було обрано наступний МК: RASPBERRY PI 2 MODEL B.

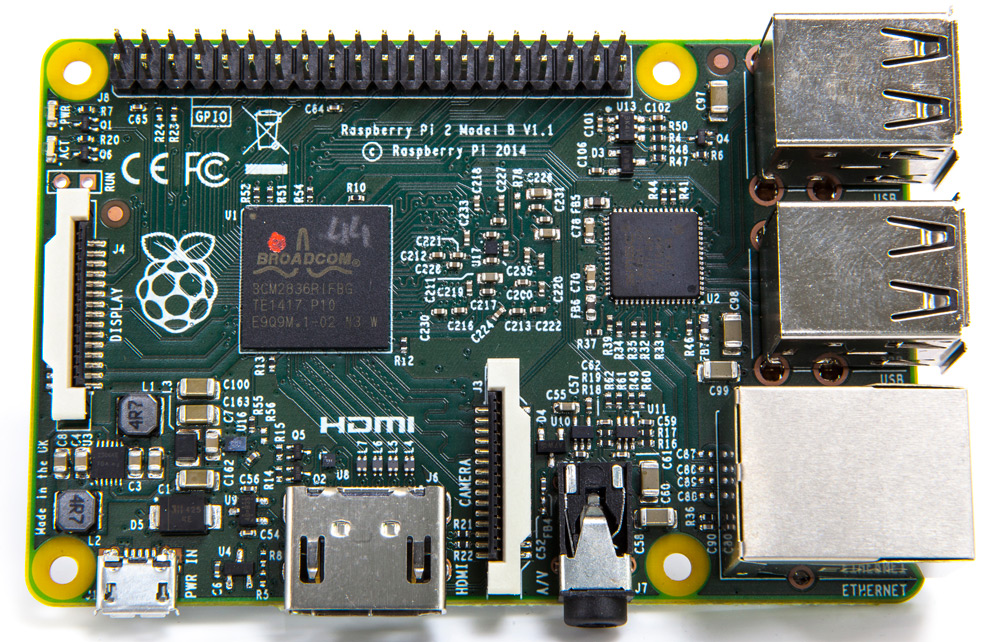


Рис. 9. Зображення Raspberry Pi 2 B

* A 900MHz quad-core ARM Cortex-A7 CPU
* 1GB RAM
* 4 USB ports
* 40 GPIO pins
* Full HDMI port
* Ethernet port
* Combined 3.5mm audio jack and composite video
* Camera interface (CSI)
* Display interface (DSI)
* Micro SD card slot
* VideoCore IV 3D graphics core

Оскільки даний МК є дуже популярним, він забезпечений суттєвою підтримкою, і тому не є великою складністю працювати над ним і використовувати його як основу для СТЗ. ОС – Linux Jessie, компілятор GNU GCC 2.6 і підтримка різних мов програмування таких як Python, C/C++ та інші.

**ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ**

На основі методу підбору оптичної системи в залежності від висоти зйомки було проаналізовано необхідні характеристики оптичної системи для ідентифікації квадрата 0.7 × 0.7 м на висоті 20 метрів. Також, було знайдена залежність розміри орієнтира ідентифікації від висоти для різних характеристик оптичного датчика. Результати представлені на рисунку 6.

На основі інформації щодо необхідних характеристик оптичного датчика була обрана камера Super WDR 800TVL Effio-V SONY CCD 28X28mm Camera Starlight 0.0003Lux.

Також, було проаналізвано мінімальні необхідні характеристики процесора обробки зображення для роботи в реальному масштабі часу. З огляду на даний аналіз було обрано як один з найкращих варінтів NVidia JETSON TK1, яка надає всі інструментальні засоби для розробки вбудованого ПО, включаючи CUDA, OpenGL 4.4 і прискорену бібліотеку OpenCV. Завдяки повному набору засобів для розробки і профілювання, а також вбудованої підтримки камер і інших периферійних пристроїв, компанія NVIDIA пропонує ідеальне рішення, яке допоможе сформувати майбутнє вбудованих систем.

1. **ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ТЕХНІЧНОГО ЗОРУ**

Найважливішими джерелами інформації про зовнішній світ для робота є його оптичні датчики і камери. Після отримання зображення необхідна його обробка для аналізу обстановки або прийняття рішення. Як було сказано раніше, комп'ютерний зір об'єднує безліч методів роботи з зображеннями.

При функціонуванні передбачається, що відеоінформація з камер обробляється будь-якої програмою, запущеної на контролері. Щоб не писати код з нуля, можна скористатися готовими програмними рішеннями. На поточний момент існує безліч готових бібліотек комп'ютерного зору.

Задача вирішується з використанням OpenCV - бібліотеки комп'ютерного зору. Вона безкоштовна як для навчальних цілей, так і для комерційного використання. Написана на оптимізованому C / C++, підтримує інтерфейси C, C++, Python, Java і включає в себе реалізації понад 2500 алгоритмів.

Крім стандартних функцій обробки зображень (фільтрація, розмиття, геометричні перетворення і т.д ...) даний SDK дозволяє вирішувати більш складні завдання, до яких відносяться виявлення об'єкта на фотографії і його «впізнавання». Слід розуміти, що завдання виявлення і розпізнавання можуть бути абсолютно різними:

* пошук і розпізнавання конкретного об'єкта;
* пошук об'єктів однієї категорії (без розпізнавання);
* тільки розпізнавання об'єкта (вже готове зображення з ним).

Для виявлення ознак на зображенні і перевірки на збіг у OpenCV є такі методи:

* Гістограма спрямованих градієнтів HOG (Histogram of Oriented Gradients) - може застосовуватися для виявлення пішоходів;
* Алгоритм Віоли-Джонса - застосовується для пошуку осіб;
* Алгоритм виявлення ознак SIFT (Scale Invariant Feature Transform);
* Алгоритм виявлення ознак SURF (Speeded Up Robust Features).

Наприклад, SIFT виявляє набори точок, які можна використовувати для ідентифікації об'єкта.



Рис. 10. Пошук об’єкта за допомогою алгоритму SIFT

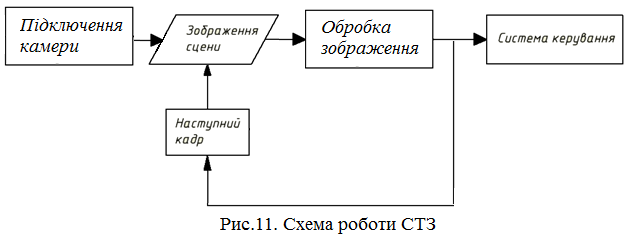
В даній роботі застосовується мова програмування Python 3.5, С++, і компілятор програмного коду GNU GCC (GNU Compiler Collection) 2.6. Дана розробка проходить в ОС Linux.

СКЗ являє собою окремою системою, програмного характеру, яка є частиною СТЗ. Саме СКЗ і є програмною складовою, яка містить в собі необхідні алгоритми роботи із зображенням, його трансформацію, обробку і знаходження цікавлячих об’єктів, а також видача сигналів на контур керування БПЛА.

* 1. **Підключення камери і настройка передачі зображення**

Перш за все потрібно налагодити обробку відеопотоку з камери в реальному масштабі часу. Відеопотік – це набір кадрів зображення місцевості, які слідують один за одним з відповідною частотою, яка залежить від характеристики камери (кадрів на секунду) і пропускною здатністю МК.

Алгоритм програмного забезпечення відео обробки наступна:



Програмне забезпечення з використанням OpenCV і Python для відеопотоку показано в Додатку А.

* 1. **Розпізнавання зображення – система комп’ютерного зору**

Розглянувши існуючи алгоритми розпізнавання зображень можна прийти до висновку щодо вибору методу розпізнавання в окремо взятому випадку.

Замість терміну “розпізнавання” часто використовують термін – “класифікація”. Постановки задач класифікації:

1. Задача ідентифікації, яка полягає у тому, щоб вирізнити певний конкретний об’єкт серед йому подібних.

2. Віднесення об’єкта до того чи іншого класу. Це може бути, наприклад, задача розпізнавання літер або прийняття рішення про наявність дефекту в деякій технічній деталі.

3. Кластерний аналіз, який полягає в розділенні заданого набору об’єктів на класи – групи об’єктів, схожі між собою за тим чи іншим критерієм. Цю задачу часто називають класифікацією без учителя, оскільки, на відміну від задачі 2, класи апріорно не задані [38].

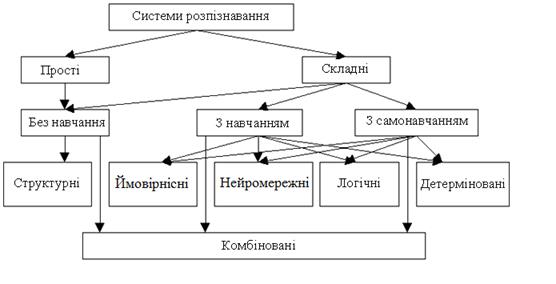


Рис. 12. Класифікація систем розпізнавання

Для дискримінантного розпізнавання, або для розпізнавання в просторі ознак, характерним є те, що кожен об’єкт зображується окремою точкою в деякому просторі. Координатними осями цього простору є ознаки, за якими здійснюється розпізнавання. Таким чином, координатами об’єктів є значення відповідних ознак. Далі, розпізнавання здійснюється на основі аналізування мір близькості між об’єктами [39].

Виділяють різні типи ознак: дихотомічні (ознака може бути присутня або відсутня; наприклад, є крила або немає крил); номінальні (наприклад, колір: червоний, синій, зелений і т.д.); порядкові (наприклад, “великий” – “середній” – “маленький”); кількісні. Для кожного типу ознак можна вводити свої міри відстані між об’єктами.

В даній роботі було використано два методи виявлення цілі на зображені сцени:

1. номінальний – пошук об’єкта на зображені сцени, знаючи його колір;
2. дихотомічний – пошук об’єкта за його специфічними ознаками (особливими точками).
   * 1. **Метод ідентифікації об’єкта за його кольором**
        1. **Загальний огляд методу**

Найпоширеніший спосіб виділити об'єкт - це колір. Колір - це властивість тіл відбивати або випускати видиме випромінювання певного спектрального складу і інтенсивності.

Однією з важливих проблем пошуку за кольором - це вплив безлічі чинників. Наприклад, освітленість. Не можна також забувати, що видимий колір - це результат взаємодії спектра випромінюваного світла і поверхні. Тобто якщо білий аркуш висвітлювати світлом червоної лампочки, то і лист буде здаватися червоним.

Трихроматична теорія (сітківка ока має 3 види рецепторів світла, відповідальних за кольоровий зір) вважає, що достатньо всього трьох чисел, щоб описати колір (червоний, синій, зелений). Тобто використовуючи три значення R, G, B.

Кольорові простору бувають лінійні і нелінійні. До лінійним відноситься RGB, нелінійні: HSV, LAB.

Зручність HSV полягає в тому, що координати обрані з урахуванням людського сприйняття:

* Hue (Тон)
* Saturation (Насиченість),
* Value (Intensity) (Інтенсивність)

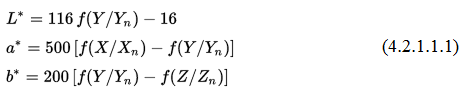
Колір, представлений в HSV, залежить від пристрою, на яке він буде виведений, так як HSV - перетворення моделі RGB, яка теж залежить від пристрою. Для отримання коду кольору, котрий залежить від пристрою, використовується модель Lab [43].

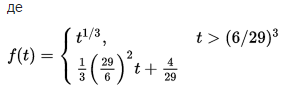
Lab - абревіатура назви двох різних (хоча і схожих) колірних просторів. Більш відомим і поширеним є CIELAB (точніше, CIE 1976 L \* a \* b \*), інших - Hunter Lab (точніше, Hunter L, a, b). Таким чином, Lab - це неформальна абревіатура, не визначальна колірний простір однозначно. Найчастіше, говорячи про просторі Lab, мають на увазі CIELAB [43].

При розробці Lab переслідувалася мета створення колірного простору, зміна кольору в якому буде більш лінійним з точки зору людського сприйняття (в порівнянні з XYZ), тобто з тим, щоб однакове зміна значень координат кольору в різних областях колірного простору виробляло однакове відчуття зміни кольору. Таким чином математично коректувалася б нелінійність сприйняття кольору людиною. Обидва колірних простору розраховуються щодо певного значення точки білого. Якщо значення точки білого додатково не вказується, мається на увазі, що значення Lab розраховані для стандартного освітлювача D50.

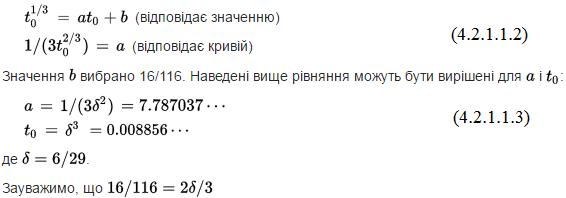
Формули визначення координат CIELAB:

1. Перетворення XYZ -> L\*a\*b\*:

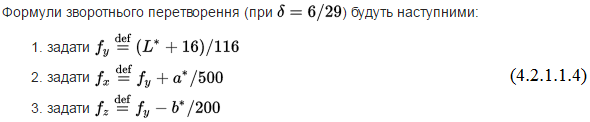


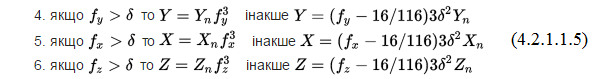


Значення  — це координати білої точки в значеннях CIE XYZ (буква n означає «нормалізована»).



1. Зворотне перетворення L\*a\*b\* -> XYZ:





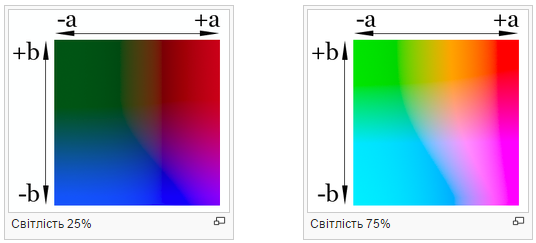


Рис. 13. Кольоровий простір LAB

Використовувати модель HSV набагато зручніше, ніж RGB, тому що координати цієї колірної моделі обрані з урахуванням людського сприйняття.

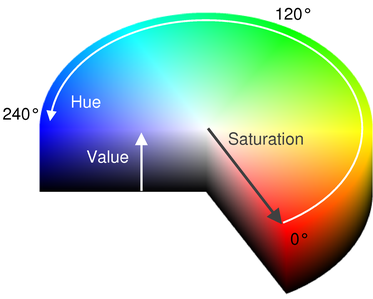


Рис. 14. Просторова модель HSV

* HSV (англ. Hue, Saturation, Value - тон, насиченість, значення) - колірна модель, в якій координатами кольору є:
* Hue - колірний тон, (наприклад, червоний, зелений або синьо-блакитний). Варіюється в межах 0-360 °, проте іноді приводиться до діапазону 0-100 або 0-1.
* Saturation - насиченість. Варіюється в межах 0-100 або 0-1. Чим більше цей параметр, тим «чистіше» колір, тому цей параметр іноді називають чистотою кольору. А чим ближче цей параметр до нуля, тим ближче колір до нейтрального сірого.

\* Value (значення кольору) або Brightness - яскравість. Також задається в межах 0-100 і 0-1.

Для виділення та пошуку об'єктів за кольором використаємо  колірну модель HSV. Даний простір кольору є нелінійним. Простір HSV (HSI) можна вважати ідеальним засобом для побудови алгоритмів обробки зображень, оскільки в його основі лежить природний опис кольору [40].

Розв’язання задачі розбивається на декілька етапів: виділення пікселів, відповідних кольору заданого об’єкту; виділення контурами знайдені об'єкти; знаходження контуру об’єкту; побудова прямокутника, в який потрапляють всі точки контуру об’єкту.

* + - 1. **Розробка програмного забезпечення алгоритму ідентифікації об’єкта за його кольором**

Лістинг коду програми для реалізації алгоритму розпізнавання об’єкту за кольором(мова програмування Python) показанов в Додатку Б.

На рис. 15 зображено об`єкт (а) та результат розпізнавання об`єкту (б) за помаранчевим кольором.

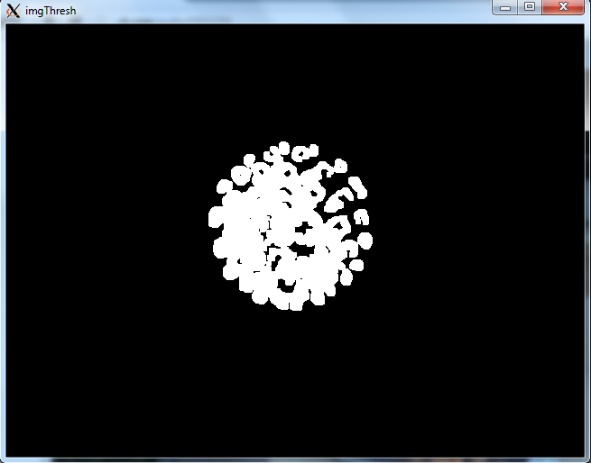
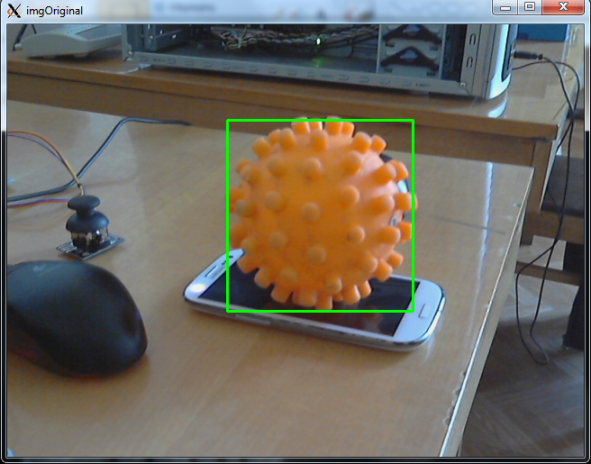


Рис. 15. Розпізнавання об'єкту за помаранчевім кольором: а) оригінал

б) маска

Після ідентифікації об'єкта знайдемо координати його геометричного центра (*x, y*) відносно оптичної осі камери в системі координат розширення камери. Знаючи початкову точку(*a, b*) зеленого прямокутника, його висоту (*h*) й ширину (*w*), знаходимо

(4.2.1.2.)

Наступним етапом є передача значень координат центра об’єкта на САК.

* + 1. **Метод ідентифікації об’єкта за особливими точками**
       1. **Загальний огляд методу**

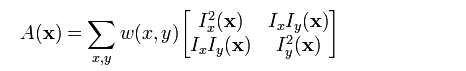
Для вирішення даної задачі застосовувався алгоритм знаходження особливих точок на зображені - SURF (Speeded Up Robust Features). В базі даних мікрокомп’ютера зберігається відповідне еталоне зображення об’єкту (символ «Н»), на якому знаходяться особливі точки за відповідним алгоритмом, а далі на кожному зображені сцени за цим же алгоритмом відшуковується наш об’єкт.

SURF є детектором і дескриптором точок інтересу в зображеннях, де зображення перетворюється в координати, використовуючи техніку піраміди мульти-дозволом, щоб зробити копію вихідного зображення з пірамідальним розподілом Гауса або форми піраміди Лапласа, щоб отримати зображення з такого ж розміру, але зі зменшеною пропускною спроможністю.

Розв’язання задачі розбивається на декілька етапів :

1. Знаходження ключових точок
2. Знаходження дескрипторів
3. Порівняння дескрипторів
4. Використання гомографії для знаходження відповідного об’єкту
5. Знаходження координат об’єкту

Перше, що слід зробити - це знайти на зображенні ключові точки. Спрощено можна вважати, що це точки в місцях різкого перепаду градієнта на зображенні по *x* і по *y* (кутові точки). Принцип їх визначення грунтується на використанні автокореляційній матриці і піраміди зображень (рис. 17) (для інваріантності до масштабу). Автокореляційна матриця складається з похідних по *x* і *y* по зображенню I.



сенс в тому, що використовувана метрика (лямбда - власні числа, det - визначник, trace - слід матриці, альфа - константа)

image

дозволяє ідентифікувати кутові точки, тому що в цих точках будуть вагомі перепади градієнтів по x і y, і R буде в локальному максимумі.

Знаходження точок інтересу

SIFT підхід використовує каскадні фільтри для виявлення масштабно-інваріантних характерних точок, де різниця гауссіанов (DOG) розраховується на перемасштабованому зображені поступово. У SURF, квадратної форми фільтри використовуються в якості наближення гауссовского згладжування. Фільтрація зображення з квадрата набагато швидше, якщо використовується інтегральний образ, який визначається як:

S(x, y)=\sum_{i=0}^{x}\sum_{j=0}^{y} I(i,j)

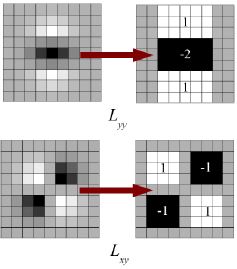


Рис. 16. Знаходження градієнтних перепадів

Сума вихідного зображення усередині прямокутника можна обчислити швидко за допомогою інтегрального зображення, що вимагає чотирьох оцінок в кутах прямокутника.

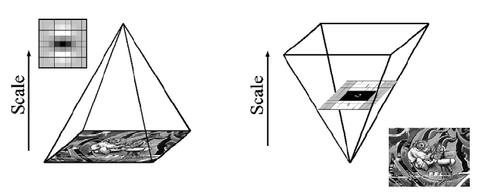


Рис. 17. Принцип масштабування в SURF алгоритмі

Далі, необхідно обчислити дескриптор – вектор, який кодує геометрію в локальній околиці навколо точки.

Локальний дескриптор околиці

Метою дескриптора є створення унікального і надійного опис функції зображення, наприклад, шляхом опису розподілу інтенсивності точок в околиці точки інтересу. Більшість дескрипторів, таким чином, обчислюється в локальній формі, отже, опис отримано для кожної точки інтересу ідентифікованої раніше.

Розмірність дескриптора має безпосередній вплив як на його обчислювальної складності та збіги точок точності. Короткий дескриптор може бути більш стійким до змін зовнішнього вигляду, але не може запропонувати достатню дискримінацію і, таким чином, дають занадто багато помилкових спрацьовувань.

В кожному блоці обчислюється домінуючий градієнтний напрямок і здійснюється поворот у бік домінуючого напрямку, як показано на рис. 18.

Перший крок складається з фіксування відтворену орієнтацію, засновану на інформації, отриманої від кругової області навколо точки інтересу. Потім побудуємо квадратну область, приведений у відповідність обраної орієнтації, і витягти дескриптор SURF з нього.

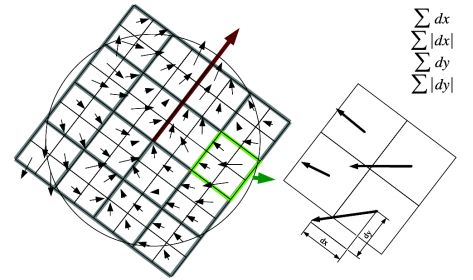


Рис. 18. Принцип знаходження дескрипторів

Крім того, існує також у вертикальному положенні версія SURF (так званий U-SURF), що не інваріантна до обертання зображення і, отже, швидше, щоб обчислити і найкраще підходить для застосування, де камера залишається більш-менш горизонтально.

Призначення орієнтації

Для досягнення інваріантності обертання, необхідно знайти орієнтацію точки інтересу. Відповіді сплеском Хаара в обох *х*- і у-напрямках в межах кругової околиці радіуса навколо точки інтересу обчислюються, де s є масштаб, в якому була виявлена ​​точка інтересу. Отримані відповіді зважуються функцією Гауса з центром в точці інтересу, то на графіку як точок в двовимірному просторі, з горизонтальною реакцією в абсцисі і вертикальної реакції в ординаті. Домінуюча орієнтація оцінюється шляхом обчислення суми всіх відповідей в ковзному вікні орієнтації розміром π/3. Горизонтальні і вертикальні реакції в межах вікна підсумовуються. Два підсумовані відповіді потім дають локальний вектор орієнтації. Найдовший такий вектор в цілому визначає орієнтацію точки інтересу. Розмір ковзного вікна є параметром, який повинен бути обраний ретельно, щоб досягти бажаного балансу між міцністю і кутовим дозволом.

На наступному етапі ми повинні «зматчіть» вектора дескрипторів, тобто знайти відповідні точки на цільовому об'єкті і в сцені. Далі ми відбираємо з усіх зматчених точок тільки ті, відстань між дескрипторами яких не більше відповідного заданого значення, який підберається експериментально для кожної задачі.

У комп'ютерного зорі будь-які два зображення одного і того ж плоского об'єкта в просторі пов'язані гомографією (якщо ми використовуємо pin-hole модель камери). Іншими словами це перетворення площина - площина. Тобто маючи набір точок на цільовому об'єкті і зіставлений йому набір точок в сцені ми можемо знайти між ними відповідність у вигляді матриці гомографії *H* (і навпаки відповідно). В основі знаходження цього перетворення лежить алгоритм RANSAC, в основі якого лежить ітеративна оцінка гомографії для випадково вибраних точок (4 на зображенні і 4 в сцені).

Далі необхідно взяти 4 точки по краях цільового об'єкта і відобразити їх за допомогою знайденого перетворення на зображенні сцени. Таким чином, ми знайдемо bounding box об'єкта в сцені. Описавши цікавлячий об’єкт геометричною фігурою, ми можемо знайти його центр.

* + - 1. **Розробка програмного забезпечення алгоритму ідентифікації об’єкта за особливими точками**

Лістинг коду ідентифікації об’єкта за особливими точками зображено в Додатку В. Код програми написаний на мові програмування С++. Нижче здійснений опис даного коду.

Перше, підключимо h файли core.hpp і opencv.hpp, що відповідають за базові класи opencv і features2d.hpp, - визначальний класи різних детекторів і дескрипторів (нас буде цікавити SURF).

#include <iostream>

#include "opencv2/opencv.hpp"

#include "opencv2/core/core.hpp"

#include "opencv2/nonfree/features2d.hpp"

#include <vector>

using namespace std;

using namespace cv;

void readme(string &message)

{ cout << message << endl; }

Далі починається тіло main, будемо вважати, що до виконуваного файлу ми передаємо 1 параметр, - шлях до картинки об’єкта (плоского об'єкта). Конструктор класу VideoCapture приймає на вхід номер камери, 0 - пристрій за умовчанням (ймовірно вбудована камера). Далі зчитується цільова картинка в img\_object.

int main( int argc, char\*\* argv )

{

VideoCapture cap(1);

if(!cap.isOpened()) // Проверка корректности отработки

{

string message = "Please check camera";

readme(message);

return -1;

}

Mat img\_object = imread( argv[1], CV\_LOAD\_IMAGE\_GRAYSCALE );

У нескінченному циклі будемо отримувати черговий кадр з камери, це є вхідне зображення сцени, в якій треба знайти цільовий об'єкт.

for(;;)

{

Mat frame;

cap >> frame; // Отримати черговий фрейм з камери

Mat img\_scene = frame;

if( !img\_object.data || !img\_scene.data ) // Перевірка наявності інформації в матриці зображення {

string message = " Error of reading the frame";

readme(message);

}

Знаходження ключових точок

Тепер приступимо до розпізнавання об'єкта. Перше, що слід зробити - це знайти на зображенні ключові точки.

// - Етап 1. Знаходження ключових точок.

int minHessian = 400;

SurfFeatureDetector detector( minHessian );

std::vector<KeyPoint> keypoints\_object, keypoints\_scene;

detector.detect( img\_object, keypoints\_object );

detector.detect( img\_scene, keypoints\_scene );

Знаходження дескрипторів

Далі, необхідно обчислити дескриптор – вектор, який кодує геометрію в локальній околиці навколо точки. В основі цього, як правило, лежить SIFT (SURF це швидкий SIFT). Принцип тут наступний:

Патч навколо даної точки розбивається на детерміновані блоки, в кожному блоці обчислюється домінуючий градієнтний напрямок і магнітуда + здійснюється поворот у бік домінуючого напрямку (інваріантність до повороту). Даний «градієнтний малюнок» описує локальний патч.

// - Етап 2. Обчислення дескрипторів.

SurfDescriptorExtractor extractor;

Mat descriptors\_object, descriptors\_scene;

extractor.compute( img\_object, keypoints\_object, descriptors\_object );

extractor.compute( img\_scene, keypoints\_scene, descriptors\_scene );

Порівняння дескрипторів

На наступному етапі ми повинні «зматчіть» вектора дескрипторів, тобто знайти відповідні точки на цільовому об'єкті і в сцені. Для цієї мети можна використовувати FlannBasedMatcher (його слід використовувати для великих наборів ключових точок) або BruteForceMatcher (навпаки). Далі ми відбираємо з усіх сматченних точок тільки ті, відстань між дескрипторами яких не більше 3 \* min\_dist, де min\_dist - мінімальна відстань між дескрипторами.

//-- Етап 3: Необхідно зматчити вектора дескрипторів.

FlannBasedMatcher matcher;

vector< DMatch > matches;

matcher.match( descriptors\_object, descriptors\_scene, matches );

double max\_dist = 0; double min\_dist = 100;

//-- Обчислення максимального і мінімального відстані серед всіх дескрипторів в просторі ознак

for( int i = 0; i < descriptors\_object.rows; i++ )

{

double dist = matches[i].distance;

if( dist < min\_dist ) min\_dist = dist;

if( dist > max\_dist ) max\_dist = dist;

}

printf("-- Max dist : %f \n", max\_dist );

printf("-- Min dist : %f \n", min\_dist );

//-- Відібрати тільки хороші матчі, відстань менше ніж 3 \* min\_dist

vector< DMatch > good\_matches;

for( int i = 0; i < descriptors\_object.rows; i++ )

{

if( matches[i].distance < 3 \* min\_dist )

{

good\_matches.push\_back( matches[i]);

}

}

Mat img\_matches;

//-- Намалювати хороші матчі

drawMatches( img\_object, keypoints\_object, img\_scene, keypoints\_scene,

good\_matches, img\_matches, Scalar::all(-1), Scalar::all(-1),

vector<char>(), DrawMatchesFlags::NOT\_DRAW\_SINGLE\_POINTS );

Використання гомографії

Використовуємо алгоритм RANSAC:

//-- Локалізація об'єктів

vector<Point2f> obj;

vector<Point2f> scene;

for( int i = 0; i < good\_matches.size(); i++ )

{

obj.push\_back( keypoints\_object[ good\_matches[i].queryIdx ].pt);

scene.push\_back( keypoints\_scene[ good\_matches[i].trainIdx ].pt);

}

Mat H = findHomography( obj, scene, CV\_RANSAC );

Беремо 4 точки, і шуканий об’єкт описуємо паралелепіпедом (bounding box) на зображені сцени.

//-- Отримати "кути" зображення з цільовим об'єктом

std::vector<Point2f> obj\_corners(4);

obj\_corners[0] = cvPoint(0,0); obj\_corners[1] = cvPoint( img\_object.cols, 0 );

obj\_corners[2] = cvPoint( img\_object.cols, img\_object.rows ); obj\_corners[3] = cvPoint( 0, img\_object.rows );

std::vector<Point2f> scene\_corners(4);

//-- Відобразити кути цільового об'єкта, використовуючи знайдене перетворення, на сцену

perspectiveTransform( obj\_corners, scene\_corners, H);

//-- З’єднати відображені кути

line( img\_matches, scene\_corners[0] + Point2f( img\_object.cols, 0), scene\_corners[1] + Point2f( img\_object.cols, 0), Scalar(0, 255, 0), 4 );

line( img\_matches, scene\_corners[1] + Point2f( img\_object.cols, 0), scene\_corners[2] + Point2f( img\_object.cols, 0), Scalar( 0, 255, 0), 4 );

line( img\_matches, scene\_corners[2] + Point2f( img\_object.cols, 0), scene\_corners[3] + Point2f( img\_object.cols, 0), Scalar( 0, 255, 0), 4 );

line( img\_matches, scene\_corners[3] + Point2f( img\_object.cols, 0), scene\_corners[0] + Point2f( img\_object.cols, 0), Scalar( 0, 255, 0), 4 );

//-- Show detected matches

imshow( "Good Matches & Object detection", img\_matches );

if(waitKey(30) >= 0) break;

}

}

//-- Кінець основного циклу

Результат роботи програми представлено нижче.

Після ідентифікації об’єкта, по аналогії з попереднім методом (за кольором), знаходимо значення координат центра, і передаємо ці значення на контур САК.

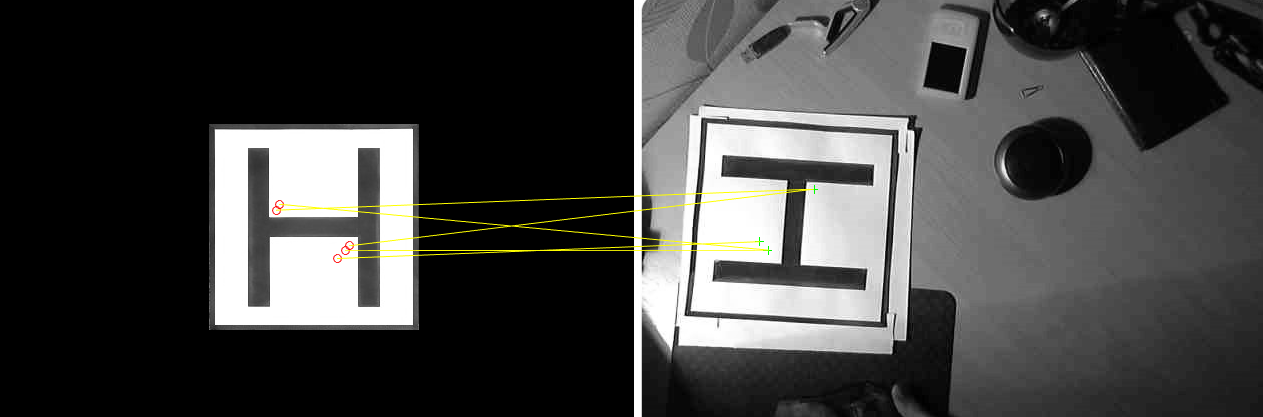


Рис. 19. Результат роботи СТЗ (в середовищі MatLAB)

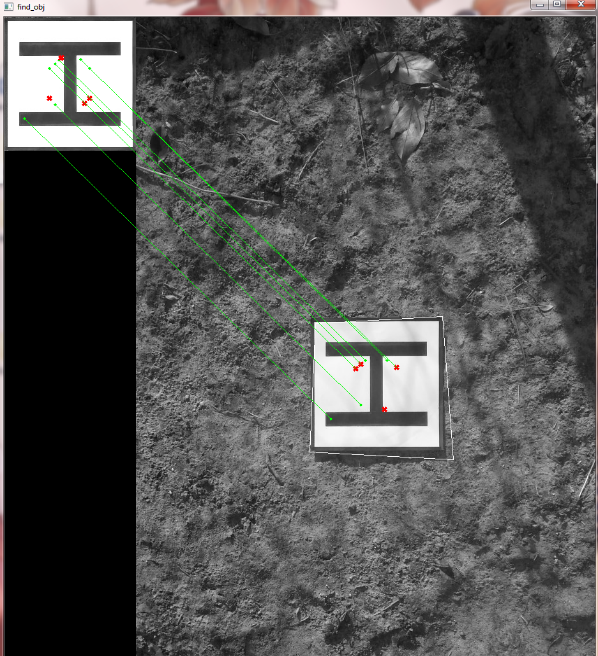


Рис. 20. Результат роботи програми СТЗ

На виході маємо координати центру знайденого об’єкта, відповідно до формул 4.1.

* 1. **Застосування фільтра Калмана в системі технічного зору з урахуванням динаміки руху БПЛА**

Представимо ситуацію, коли СТЗ ідентифікувала орієнтир та зафіксувала його положення відносно оптичної осі камери. Зрозуміло, що квадрокоптер має свою певну динаміку руху, яка відома, щоб провести аналіз, з якою швидкістю може змінюватись положення оріентира на зображені. Тому при раптовій втраті орієнтира із зображення можна дізнатися, чи дійсно орієнтир зник з площини зйомки. Приклад втрати орієнтира зображена на рисунку 21. Червоний колір показує знайдений орієнтир.

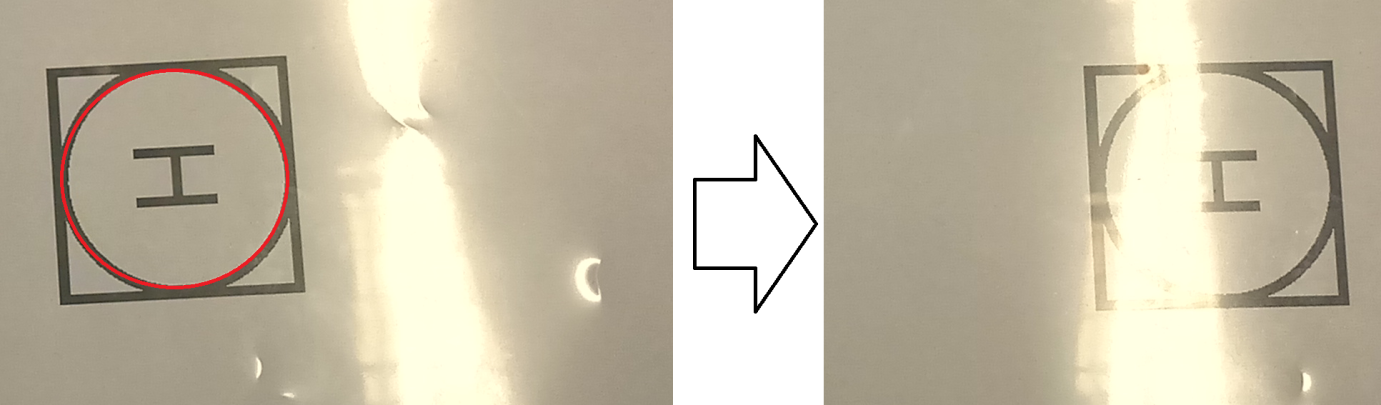


Рис. 21. Вплив освітлення навколишнього середовища на розпізнавання орієнтирів.

Причинами раптових втрат орієнтира із зображення бувають різними: умови зйомки (освітлення, тінь від хмар, тощо) чи сторонній об‘єкт, який потрапив між камерою та орієнтиром. На основі вимірювань від ІНС ми можемо дізнатися, на скільки чи як було змінено положення квадрокоптера і зробити висновок, чи можливо при такому переміщені було втратити орієнтир із зображення. Для вирішення цієї проблеми було застосовано фільтр Калмана, який на основі вимірювання з ІНС (орієнтація та навігація БПЛА) зпрогнозує, чи дійсно орієнтир був втраченний із зображення камери.

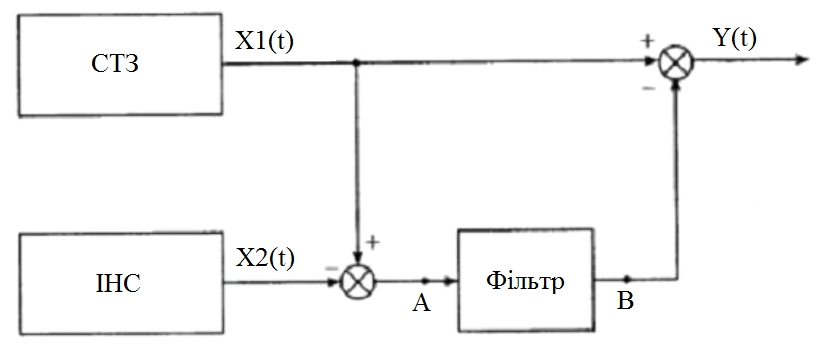


Рис. 22. Структурна схема комплексування за компенсаційним способом.

* + 1. **Синтез фільтра Калмана комплексної системи орієнтації та навігації**

З теоретичної точки зору, фільтр Калмана є тим, що називається лінійно-квадратичною проблемою, якою є задача оцінки миттєвого значення лінійної динамічної системи збудженої білим шумом з вимірювань, які лінійно зв‘язані з станом системи, але є пошкодженими білим шумом [44, 45, 46, 47, 48]. Отримана оцінка, являє собою статично оптимальну по відношенню до будь-якої квадратичної фукції похибки оцінки [44, 47, 48].

З практичної точки зору, фільтр Калмана являється одним із великих відкриттів в історії статичних теорій оцінювання. Найбільного використання фільтрів набув в керуванні складних динамічних систем, такі як неперервні процеси виробництва, в літаках, кораблях та космічних апаратах [48, 49, 50, 46].

При використанні фільтра Калмана динамічна система повинна бути представлена у вигляді моделі динаміки і моделі вимірювання, відповідно до структури фільтру [44, 45, 50]. Для кожного такту *k* роботи фільтру необхідно у відповідністі до наведеного нижче опису визначити матрицю еволюції процесу , матрицю спостерігання , коваріфційну матрицю процесу , коваріаційну матрицю шуму вимірювань ; при наявності керуючих впливів – матрицю їх коефіцієнтів [44, 51, 45, 50].

Модель процесу передбачає, що істинне значення у момент *k* визначається з істинного стану у момент (*k* – 1) у відповідності до рівняння [44, 51, 45, 50]:

 (4.3.1)

де – вектор детермінованих керуючих впливів, – нормальний випадковий процес з нульовим математичним очікуванням та коваріаційною матрицею , яка потребує визначення дисперсії процесу [44, 45, 48].

Результат спостереження (вимірювання) істинного вектору стану у момент *k* зв‘язаний з ним рівнянням:

 (4.3.2)

де – нормальний випадковий процес з нульовим математичним очікуванням і коваріаційною матрицею , описуючий білий шум у результатах вимірювання [44, 45].

Початковий стан та вектори випадкових процесів на кожному етапі тракту  вважаються незалежними.

В загальному випадку фільтр Калмана працює в два єтапи: прогноз і корекцію [45, 50]. Оцінка фазового стану системи, яка була отримана на попередньому кроці роботи розповсюджується на поточний крок *k* у вигляді прогнозу оцінки  відповідно до моделі динаміки системи (4.3.1). Після цього, отримана оцінка та вимірювання, які були отримані на даному кроці,  використовують для отримання відкорегованої оцінки стану системи [45].

При цьому прогнозування можуть відбуватися безліч разів між двома корекціями. Це використовується для розмовсюдження фазового стану, коли вимірювання за якимось причинами виявляються недоступними або недостовірними [52, 53, 45, 50].

Похибка оцінки стану системи описується в термінах коваріаційної матриці похибки [44, 45, 50]:

, (4.3.3)

де це математичне очікування.

Робота фільтру починається з ініціалізації оцінки вектора стану і коваріаційної матриці похибки [44, 51, 45, 50]:

 (4.3.4)

 (4.3.5)

Далі, відбувається розповсюдження оцінки стану і матриці коваріацій відповідно до моделі (4.3.1) для отримання некоректованої оцінки [44, 45, 48]:

 (4.3.6)

 (4.3.7)

На наступному кроці визначається коефіцієнт підсилення (підсилення Калмана) та відбувається корекція оцінки з урахуванням отриманих вимірювань [44, 51, 45, 50]:

 (4.3.8)

 (4.3.9)

 (4.3.10)

Описаний фільтр Калмана є орієнтований на вирішення лінійних моделей, але існує декулька різновидів фільтру, які мають за мету вирішення нелінійних задач. Найбільш розповсюдженим серед них є розширений фільтр Калмана. По своїй стректурі він схожий на лінійну версію, але має відмінність у тому, що в рівняннях динаміки присутні нелінійні функції [44, 51, 45, 50].

* + 1. **Реалізація фільтру Калмана**

Відомо, що ідентифікований орієнтир на зображені (координатна сітка в пікселях з осями Х та У) рухається за наступними рівняннями:

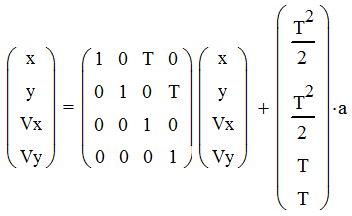
де (*x, y*) – координати орієнтира, () – швидкість зміни положення орієнтира, () – прискорення зміни положення орієнтира, *t* – час.

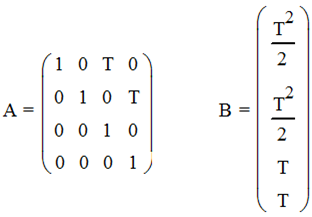
Зрозуміло, що прискорення зміни положення орієнтира залежить від динаміки польоту квадрокоптера, тому вважаємо, що наш керуючий вплив –прискорення зміни положення квадрокоптера відносно орієнтира в момент часу *k*.

Для синтезу фільтра Калмана, запишемо дані рівняння як модель процесу:

та як рівняння вимірювання:

Будемо вважати, що Т – час між циклами обробки зображення (час дискретизації). Тоді для орієнтира модель процесу є наступна:

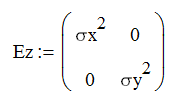




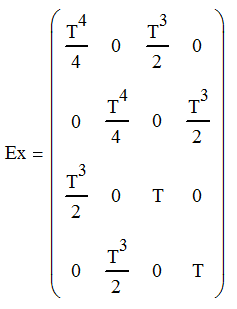
Оскільки здійснюємо вимірювання тільки по координатам, то матриця спостереження буде мати наступний вид:



Далі запишемо так звані помилки системи гаусівського розподілу. Похибка вимірювання прийме наступний вигляд:



Коефіцієнти помилок мають своє джерело від похибок СТЗ вимірювання положення орієнтира, які в свою чергу залежать від відстані розпізнавання орієнтира та якості зображення. Для нашого експерименту дані коефіцієнти прирівняємо до значення «1» - похибка зміщення по пікселю.



Маючи всі необхідні матриці для застосування фільтру Калмана, записуємо його алгоритм в середовищі програмування Matlab – Додаток Г. В даному додутку також записані коментарі пояснення щодо розробки коду.

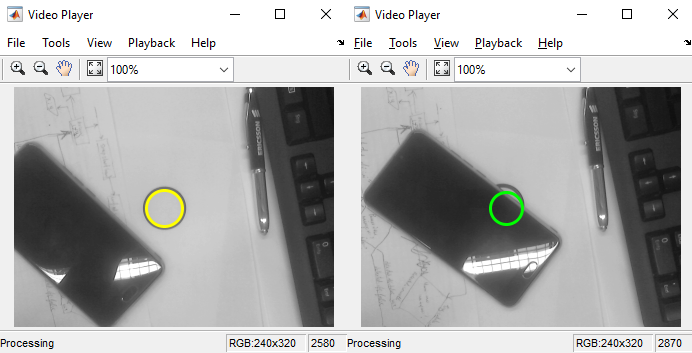


Рис. 23. Результат роботи СТЗ з фільром Калмана: жовте коло – результат роботи СТЗ, зелене – прогнозування за фільтром Калмана

Як результат маємо алгоритм, який дозволяє при втраті орієнтира протягом наступних декількох секунд в залежності від динаміки руху БПЛА прогнозувати позицію орієнтира.

**ВИСНОВКИ ДО ЧЕТВЕРТОГО РОЗДІЛУ**

В даному розділі була проаналізована структура, принцип та методика роботи системи комп‘ютерного зору як частина системи технічного зору, а також її техніка розробки в ОС Linux з використанням різних мов програмування з засобами OpenCV та Matlab.

Було описано алгоритми роботи декілької методів ідентифікації орієнтира, які відрізняються між собой принциповою методикою обробки зображення. Також, на основі даних принципів, було розроблено програмне забезпеченнясистеми комп‘ютерного зору на базі двух принципів ідентифікації орієнтира: за його кольором та за точками інтересу. Як результат, було продемонстровано роботи даного ПО на рисунках 15, 19 та 20.

Для підвищення надійності системи технічного зору було впроваджено фільтр Калмана. З огляду на те, що ми маємо параметри та динаміку рузу БПЛА, даний фільтр є невід‘ємною частиною в комбінованій системі керування БПЛА, оскільки ми можемо зпрогнозувати, де може знаходитись знайдений раніше орієнтир, якщо він зник із зображення. Реалізація фільтра Калмана в даній роботи представлено в розділі 4.3.2, а результат його роботи показано на рисунку 23.

1. **МОДЕЛЮВАННЯ ДИНАМІКИ ПОЛЬОТУ БПЛА ТИПУ КВАДРОКОПТЕР**
   1. **Розробка і аналіз рівнянь квадроторної системи**

Для розробки точних систем керування для платформи квадрокоптера необхідно розробити та проаналізувати рівняння руху, що визначають систему квадрокоптера. Квадрокоптер визначається набором нелінійних рівнянь, які роблять важким точне моделювання та контроль. Розробивши всі компоненти рівнянь, можна спростити рівняння руху після прийняття декількох припущень щодо методу, який буде експлуатувано в квадрокоптері. Мета полягає в тому, щоб створити модель, яка є простим, але настільки реальною, наскільки це можливо.

* + 1. **Змінні та координатне визначення системи**

Зміна положення квадрокоптера працює на концепції змінних моментів і ударів. Кожен ротор складається з безщіткового DC мотора та ротора з фіксованим кроком. Електродвигуни розташовані по парам по горизонтальній та по вертикальній осях, де одна пара рухається за годинниковою стрілкою, а горизонтальна пара обертається проти годинникової стрілки. Така конструкція двигунів призводить до моментів реакції, оскільки пара двигунів точно протилежні один одному, якщо всі вони обертаються з однаковою швидкістю. Усунення обертального моменту дозволяє дрону підтримувати постійну рух при русі. Контроль двигуна здійснюється шляхом зміни швидкості пар двигунів, щоб створити відцентровий момент, що не відповідає нулю. Висота контролюється, змінюючи зусилля від кожного двигуна рівними кількостями, щоб забезпечити чистий вектор тяги без моменту обертання. Для переміщення в бічних напрямках, відносна швидкість кожного двигуна у бічній парі змінюється для створення бажаного бічного зміщення. Простий дизайн такого квадрокоптера складається з невеликих платформ, що працюють від батарей, здатні виконувати стабільне наведення, і безпечні для використання в приміщенні. Квадрокоптер класифікується як система з недостатньою активністю. Хоча квадрокоптер може рухатися в 6 ступенів свободи (3 поступальні і 3 обертальні), існує всього 4 входи, які можна контролювати (швидкість 4 двигунів). Як буде показано нижче, обертова та поступальна динаміка пов'язані, що представляє цікаву проблему керування [61].

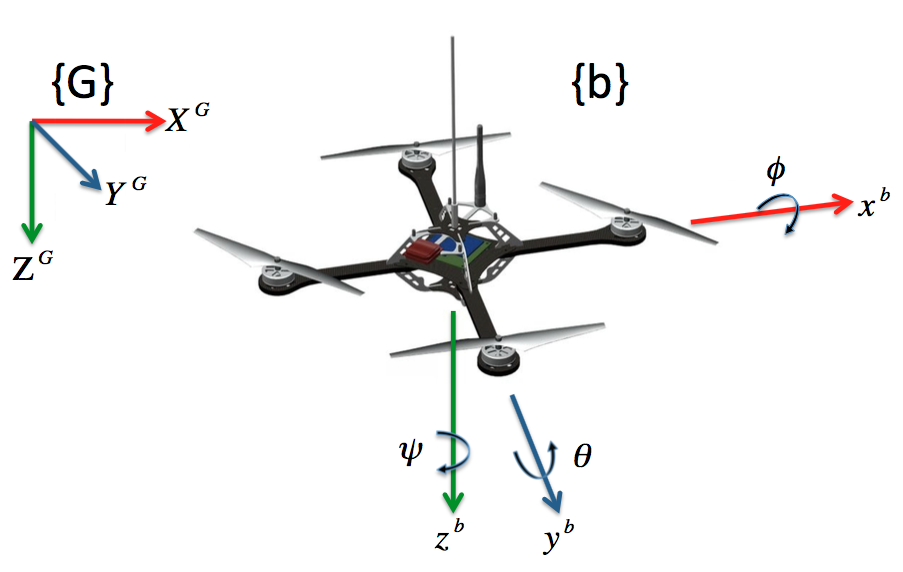


Рис. 25 Координатне визначення квадроторної системи

Основна квадрокоптерна структура, що використовується для розробки моделі, показана на рисунку вище, що зображує кути Ейлера з креном, тангажом, рисканням, систему координат тіла {b} та глобальну систему координат {G}. Дана модель спирається на кілька припущень [70]:

* Квадрокоптерна конструкція є жорсткою і симетричною з центром маси, вирівняною з центром каркаса квадрокоптера.
* Повштох і сила тяги кожного двигуна пропорційні квадрату швидкості двигуна.
* Пропелелери вважаються жорсткими, і тому деформація лопатей гвинтів незначні (не враховуються).
* Земля плоска і не обертається.
* Збурюючі моменти від навколишнього середовища є незначними (враховується як помилка між бажаним та реальним положенням).
* Наземний ефект незначний.

Повний перелік змінних стану показано нижче. Ця позначка використовує позначення, взяті з літератури з аеронавтики, зокрема, координатну систему Північ, Схід, Південь (ПСП). Позиція квадрокоптера дається в глобальній СК, тоді як швидкість і кутова швидкість визначаються в будівельній СК квадрокоптера. Важливо зазначити, що різні загальні датчики здійснюють вимірювання в обох системах координат. GPS, наприклад, вимірює положення квадрокоптера та швидкість обертання в глобальній СК, тоді як акселерометри та гіроскопи - в СК дрона. Для більш продвинутої навігації додано рух по траєкторії шляхових точок, які задані у глобальній системі координат [62].

глобальне (північне) позиціювання квадрокоптера [м]

глобальне (східне) позиціювання квадрокоптера [м]

глобальне (південне) позиціювання квадрокоптера [м]

локальне (північне) позиціювання квадрокоптера [м]

локальне (східне) позиціювання квадрокоптера [м]

локальне (південне) позиціювання квадрокоптера [м]

*φ* кут крену вздовж осі (рад)

*θ* кут тангажу вздовж осі (рад)

*ψ* кут рискання вздовж осі (рад)

*p*  кутова швидкість вздовж осі (рад/с)

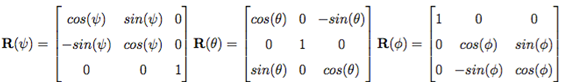
*q* кутова швидкість вздовж осі (рад/с)

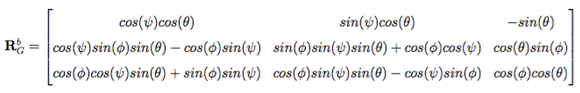
*r* кутова швидкість вздовж осі (рад/с)

* + 1. **Кінематика переходу**

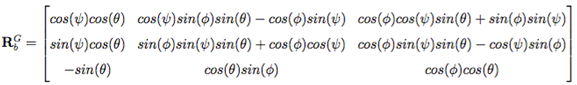
Змінні стану для швидкості знаходяться в будівельній СК дрона, але змінні стану для позиції знаходяться в глобальній системі [64]. Тому необхідно визначити матрицю обертання для перетворення змінних між системами координат. Нижче описано перетворення між глобальною та будівельною СК [54].





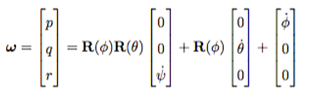


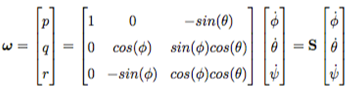


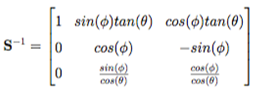


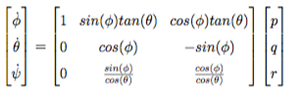
* + 1. **Кінематика обератльного руху**

Оскільки кутові швидкості визначаються в будівельній СК квадрокоптера, а кути Ейлера визначаються в проміжних СК, ми можемо використовувати матрицю перетворення, отриману вище, для визначення співвідношення між кутоврю швидкістю та похідними від кутів Ейлер за часом як показано нижче. Кутові швидкісті - це вектори, що направлені уздовж кожної вісі обертання і не є рівними похідним від кутів Ейлера за часом. Виведення нижче припускає, що похідна від часу кожної швидкості Ейлера невелика.









Якщо передбачається, що кут Ейлера малий (близько 0), то матриця S стає матрицею тотожності, а кутова швидкість приблизно дорівнює похідним від кутів Ейлера за часом.

* + 1. **Рівняння руху безщіточного двигуна**

Квадрокоптери використовують різні типи безщіткових двигунів, щоб генерувати тягу та моменти, необхідні для керування. Електронні регулятори швидкості (ESC) отримують бажаний показник швидкості двигуна від контролера польоту та надсилають ці команди кожному окремому двигуну. Наведено основну діаграму вільного тіла, яка зображує тяги та сили кожного двигуна на квадрокоптері. Зверніть увагу, що ось z була визначена за напрямком гравітації та протилежному напрямку векторів двигуна [63].

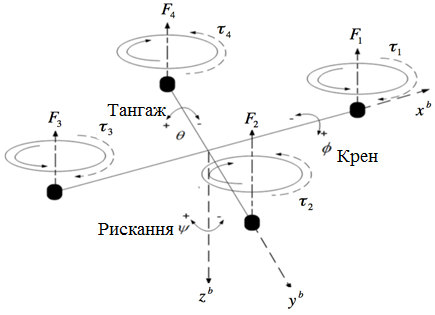


Рис 26

Перша частина рівнянь нижче дає рівняння крутного моменту та потужності типового безщіткового двигуна. Ми припускаємо, що опір двигуна та струм без навантаження становить приблизно 0. Оскільки всі двигуни однакові на квадрокоптері, для кожного двигуна застосовується одна модель. Ми одержуємо рівняння для кількості потужності, який двигун виробляє в одному спрощеному рівнянні.

*τ* крутний момент двигуна [Н м]

– пропорційна константа крутного моменту

*I* – вхідна сила струму [А]

сила струму без нагрузки [А]

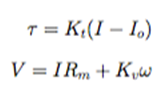
*V* – падіння напруги на двигуні [В]

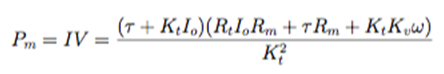
опір двигуна [Ом]

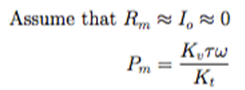
звороний EMF коефіцієнт

*ω –* кутова швидкість двигуна [рад/с]

потужність двигуна [Вт]







Наступні рівняння використовують рівняння потужності та крутного моменту, отримані вище, для розробки узагальненого рівняння тяги. Використовувана потужність під час наведення курсу - це ідеальна сила, необхідна для створення тяги, рухаючись в колоні повітря. Ми припускаємо, що швидкість руху дрона є низькою, тому швидкість при наведенні - це швидкість руху повітря при наведенні. З цього рівняння очевидно, що більші пропелери призводять до зменшення використовуваної потужності. Хоча кожен ротор залишає його після виклику при русі, припускаємо, що ця швидкість вільного потоку повітря дорівнює нулю. Це призводить до того, що тяга кожного двигуна пропорційна виробленню квадрата кутової швидкості валу ротора та коефіцієнта підйому. Це рівняння відображає співвідношення між швидкістю двигуна і результуючою силою тяги, що є основою стабільного керування транспортним засобом [66].

потужність наведення курсу [Вт]

потужність двигуна [Вт]

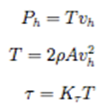
тяга наведення курсу [Н]

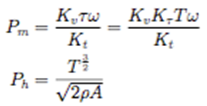
індукована швидкість наведення курсу [м/с]

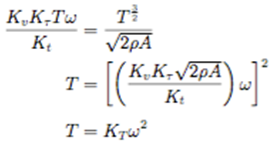
кутова швидкість двигуна [рад/с]

*ρ* – густина повітря [кг/]

*А* – площина захоплення лопатями ротора []







* + 1. **Динаміка лінійного руху**

Лінійні рівняння руху визначаються в глобальній системі відліку. Прискорення квадрокоптера в глобальній системі координат дорівнює сумі сили тяжіння, сили тяги двигунів та лінійної сили тертя, в результаті яких здійснюється переміщення. Вектори тяги у СК тіла перетворюються в глобальну СК за допомогою матриці перетворення, визначеної раніше. Результуюча сила, створювана чотирма двигунами в глобальній СК, визначена нижче. Крім того, результуюча глобальна сила опору по кожній осі визначається нижче як коефіцієнти переміщення та лінійних швидкостей у кожному напрямку у глобальній системі. Це проста модель тертя рідини. Більш складне моделювання включатиме окремі константи тертя для кожної осі, а також моделювання тертя в СК тіла замість інерційної системи [67].

*m* – маса квадрокоптера [кг]

*g* – земне прискорення [м/]

прискорення в глобальній системі координат [м/]

швидкість в глобальній системі координат [м/]

сила тяжіння в глобальній системі координат [Н]

сила переміщення в глобальній системі координат [Н]

загальна сила тяги в глобальній СК [Н]

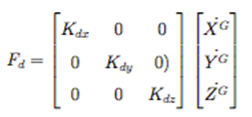
загальна сила тяги в звязній СК [Н]

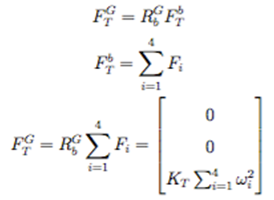
матриця переходу зі звязаної СК в глобальну СК [Н]

тяга кожного двигуна [Н]









* + 1. **Динаміка обертального руху**

Рівняння обертального руху визначаються в СК тіла таким чином, що обертання можна обчислити по відношенню до центру квадрокоптера, а не до центру глобальної СК. Оскільки ми передбачали, що квадрокоптер буде симетричним, квадратний момент інерційної матриці буде симетричним [68]. Ці рівняння складаються з трьох умов, включаючи аеродинамічні ефекти та моторні моменти. По-перше, гіроскопічний ефект, зумовлений поворотом твердого тіла. По-друге, двигуни виробляють обертання, вібрації та крутний момент. Показано, що крутний момент за рахунок переміщення пропорційний товщині константи та квадрату кутової швидкості ротора. Це рівняння припускає, що квадрокоптер працює в стабільному польоті і що пропелери підтримують постійну тягу, а не прискорюють. Це припущення призводить до того, що крутний момент щодо глобальної осі z дорівнює крутному моменту, викликаному переміщенням. По-третє, перехресний зв‘язок, який описує гіроскопічний ефект від повороту гвинта, пов'язаний з обертанням тіла. Це пов'язано з тим, що вісь обертання сама рухається з кутовою швидкістю рами [69].

*ω* – кутова швидкість [рад/с]

– кутове прискорення [рад/]

момент інерції квадрокоптера [кг/]

момент інерції двигуна [кг/]

момент інерції ротора [кг/]

момент інерції квадрокоптера по осі *X* [кг/]

момент інерції квадрокоптера по осі *Y* [кг/]

момент інерції квадрокоптера по осі *Z* [кг/]

кутовий момент від двигунів [Н/]

кутовий момент від гіроскопічного ефекту [Н/]

кутовий момент по відповідних осях [Н/]

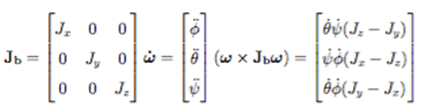
*R* – радіус пропелера []

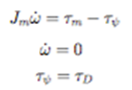
*A* – перетин пропелера []

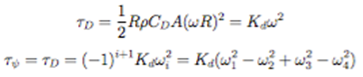
безрозмірний коефіцієнт

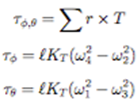
*l* – довжина плеча []

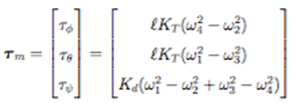


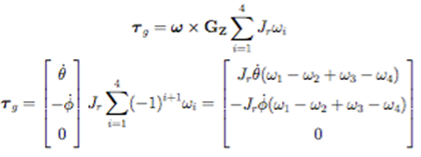






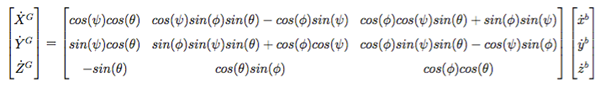


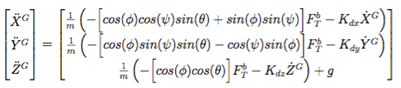


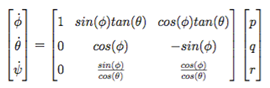


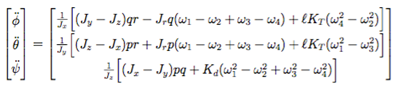
* + 1. **Рівняння руху**

Вирішення для глобального поступального прискорення квадрокоптера та кутових прискорень тіла приводить до наступних рівнянь. Ми використовуємо глобальну СК для трансляційної позиції, оскільки вона є такою ж координатною рамкою, як наш GPS-датчик. Подібним чином, рама тіла вибирається по відношеню до ІНС (акселерометр, магнітометр і гіроскоп), які також проводять вимірювання в СК тіла. Ці рівняння є як нелінійними, так і сильно зв'язаними. Їх можна спростити, використовуючи обґрунтовані припущення для розробки та впровадження стабільних систем керування [60].









* 1. **Оцінка параметрів моделі, основані на нашій фізичній системі**

Даний розділ спирається на розділ 5.1, описаний вище. У цьому розділі ми оцінюємо параметри моделі на основі нашої фізичної системи. Математична модель всіх нижче наведенних розрахунків описується у роботах [81, 82, 83].

Це не тільки покращить точність нашої симульованої моделі, але також допоможемо визначити точні параметри для наших систем керування. Чим точніше початкові параметри керування, тим менше часу, необхідного для проведення початкової виставки.

* + 1. **Параметри фізичної моделі**
       1. **Момент інерції квадрокоптера**

Ці параметри засновані на комплектації 3DR ArduCopter Quad-C Frame DIY [55]. Моменти інерції можна розрахувати кількома способами. У рівняннях нижче, складання рами моделюється порожньою сферою. Стійка - це центр квадрокоптера, який тримає автопілот APM 2.6+, 3DR uBlox GPS та компас, акумулятор 4000mAh 3S 30C LiPo, модуль живлення APM та розподільна плата живлення. Також ми додаємо вагу чотирьох ESC 20А, радіоприймача Spektrum AR8000 [56] та телеметричний радіоприймач 3DR 433МГц. Ми моделюємо двигуни 880Kv як циліндри та використовуємо метод Гюйгенс-Штайнер, щоб перемістити осі відліку циліндрів до сфери. Ці рівняння, а також відповідні значення обчислюються нижче. Зверніть увагу, що квадрокоптер можна моделювати за допомогою й інших геометричних фігур [57].

маса стеків рами та електроніки [кг]

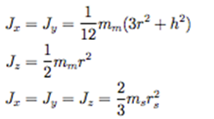
радіус стеків [м]

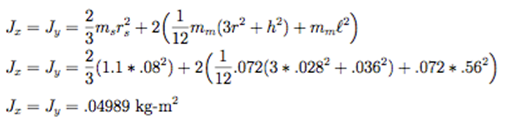
маса моторів [кг]

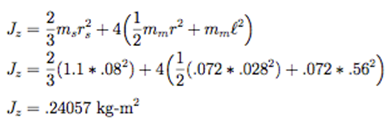
*r* – радіус мотора [м]

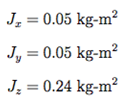
*h* – висота мотора [м]

*l* – довжина плеча [м]









* + - 1. **Момент інерції пропелера**

Щоб обчислити момент інерції пропелерів, можна використовувати роботу магістерської дисертації Томаса Цзінека [58]. Зокрема, в розділі 4.1.1 обчислюється двигун інерції для пропелера APC 10 × 4,7, який майже ідентичний пропелер APC 11 × 4,7, використовуваним для цього проекту. Він вимірює вагу і щільність пропелера, ділить пропелер на секції і оцінює загальний момент інерції як суму розрізів, використовуючи теорему про паралельні осі. Він оцінює момент інерції, рівний 0,04439 кг-м^2.

* + 1. **Характеристика датчиків**

Нижче наведені датчики стандарту з автопілоту APM 2.6+. Для того, щоб поліпшити наше моделювання і, отже, дизайн контролера, ми повинні точно імітувати роботу датчиків. Ці параметри будуть використані для додавання помилок до вимірювань імітації датчиків. Потім ми можемо використовувати фільтр, щоб зменшити вплив шуму датчика, а також об'єднати різні вимірювання датчиків для розрахунку більш точної інформації щодо положення та орієнтації. Даний електронний ресурс [59] має релевантні параметри для інших типів датчиків, які не розглядаються нижче. Нижче наведена інформація про таблицю виробника, після чого обговорюється, як вручну оцінювати параметри датчика.

3DR uBlox LEA-6 GPS з комплектом цифрового компаса HMC5883L (GPS Datasheet [71], Compass Datasheet [72]):

* GPS: швидкість оновлення 5 Гц, 2,5 м CEP
* Компас: 160 Гц Максимальна вихідна оцінка, 1 ° до 2 ° точність курсу, чутливість 4.3 мГ / digit

MPU-6000 6-осі гіро + Accel (специфікація [73])

* Гіроскоп: максимальна вихідна оцінка 8 кГц, чутливість 16,4 ЛСВ / сек / сек, шум 0,05 / сек
* Акселерометр: максимальна вихідна оцінка 1 кГц, чутливість 2,048 літр / с, гучність шуму 400 мкг / √Hz

MS5611-01BA03 Датчик барометричного тиску (таблиця даних [74])

* Максимальна вихідна оцінка 2kHz, роздільна здатність 10 см, +/- 0.5 Гпа відносної точності.
  + - 1. **Моделювання роботи гіроскопа**

MEMS гіроскоп містить невеликий вібрації. Для виявлення повороту вимірюється зміна частоти платформи через ефект Коріоліса при повороті платформи. Якщо гіроскопи вирівняні по осям x, y та z квадрокоптера, то гіроскопи вимірюють кутові коефіцієнти тіла p, r та q.

Гіроскоп можна моделювати як комбінацію справжнього вимірювання, зміщення нуля та наявність шуму. Справжня кутова швидкість вимірюється в радіанах на секунду. Шум характеризується як нульовий середній гауссовий шум. Упередженість залежить від температури і може бути оцінена для кожного польоту. Ці елементи дають наступні рівняння для моделювання гіроскопа. Ця ж модель використовується для кожної осі [81, 82, 83].

виміряна кутова швидкість вздовж осі [рад/с]

виміряна кутова швидкість вздовж осі [рад/с]

виміряна кутова швидкість вздовж осі [рад/с]

істинна кутова швидкість вздовж осі [рад/с]

істинна кутова швидкість вздовж осі [рад/с]

істинна кутова швидкість вздовж осі [рад/с]

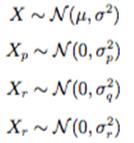
*β* – виміряне зміщення [рад/с]

*X –* рандомна змінна балого шуму [рад/с]

*σ* – стандартне відхилення [рад/с]

*μ* – середнє [рад/с]

*t* – час [с]





Щоб оцінити зміщення та стандартне відхилення білого шуму, IMU було запущено на частоті 100 Гц протягом 10 хвилин. Результати були записані у файлі .csv, використовуючи цей файл Arduino.pde для запису даних датчика. Далі був запущений сценарій Matlab для обчислення шумових характеристик, а також для побудови записаних вимірювань гіроскопа та моделей вимірювань гіроскопа. Обидва ці виходи показані нижче.

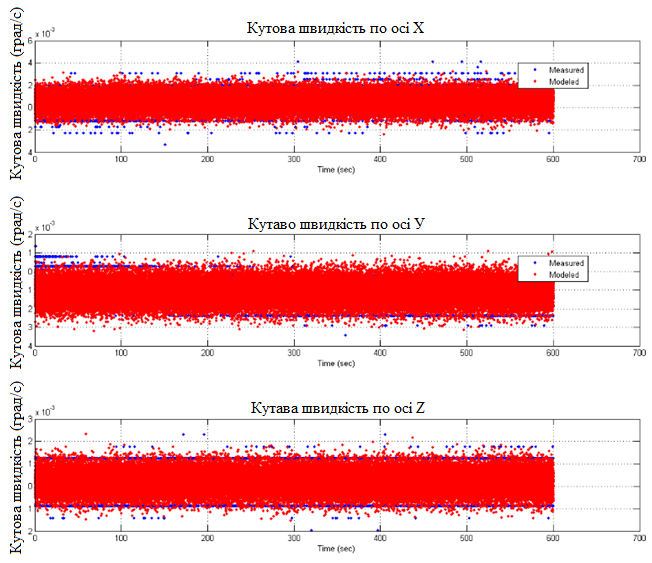


Рис. 27. Порівняння виміряної та промоделбованої роботи гіроскопа

* + - 1. **Моделювання акселерометра**

MEMS акселерометр містить торсіонні важелі, прикріплені до маленької пластини. Пластина обертається під час прискорення, що змінює ємність між пластиною та навколишніми стінками. Якщо акселерометр вирівняний по осям x, y та z квадрокоптера, то вихід акселерометра - лінійне прискорення в СК тіла.

Акселерометр можна моделювати як комбінацію справжнього вимірювання, зміщення нуля та шуму. Справжнє лінійне прискорення вимірюється в метрах на секунду в квадраті. Шум характеризується як нульовий середній гауссовий шум. Упередженість залежить від температури і може бути оцінена для кожного польоту. Ці елементи дають наступні рівняння для моделювання акселерометра. Ця ж модель використовується для кожної осі [81, 82, 83].

виміряне прискорення вздовж осі [м/]

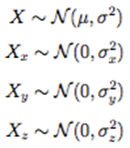
виміряне прискорення вздовж осі [м/]

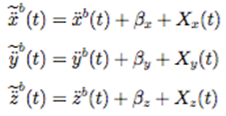
виміряне прискорення вздовж осі [м/]

виміряне прискорення вздовж осі [м/]

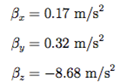
виміряне прискорення вздовж осі [м/]

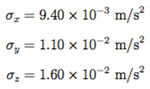
виміряне прискорення вздовж осі [м/]





Шумові характеристики акселерометра оцінювалися так само, як і гіроскоп. Вимірювання були записані на частоті 100 Гц за допомогою Arduino та після обробки в Matlab. Нижче наведено характеристики шуму та порівняння значень вимірювання та значень моделювання прискорення.





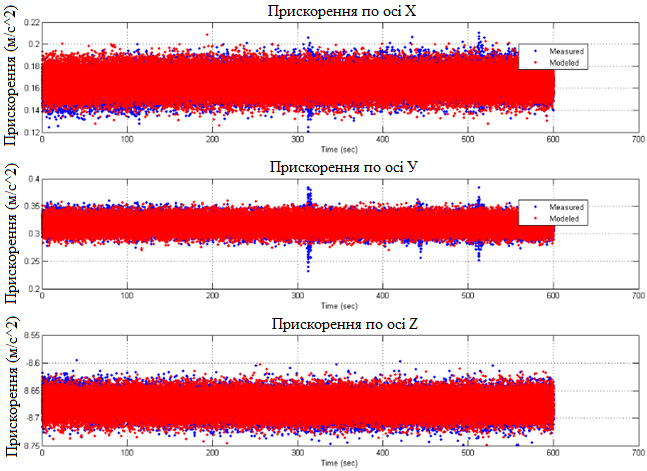


Рис. 28. Порівняння виміряної та промоделбованої роботи акселерометра

Як для гіроскопа, так і для акселерометра ми спростили розробку моделі. По-перше, кожен датчик був нерухомим при здійснені вимірювання. Зрозуміло, що ці датчики будуть постійно рухатися. Також датчики вимірювалися при постійній температурі. Насправді зміна висоти призведуть до зміни температури, що вплине на зміщення нуля у датчиків. Обидва ці чинники не враховуються в цій простій моделі.

* + 1. **Параметри системи двигуна**

Безщіткові двигуни постійного струму, такі як чотири двигуна 880Kv, що використовуються на цьому квадрокоптері, популярні у квадротерів, оскільки вони є невеликими двигунами з високим крутним моментом. Ці двигуни знаходиться зовні, а не всередині. Це дозволяє двигунам генерувати вищий крутний момент. Цей високий крутний момент до вагового співвідношення дає високий ефективний коефіцієнт, який потребує менше ваги акумулятора та більше часу польоту між заряджаннями. Великі моменти дозволяють моторам швидше змінювати швидкість, що в кінцевому підсумку керує рухом квадрокоптера.

Основні фізичні принципи, що використовуються для опису традиційного мотора постійного струму, можна застосувати до безщітного мотора постійного струму. Спрощене електричне рівняння (Закон Кірхгофа) та динамічне рівняння мотора, пов'язаного з навантаженням (другий закон Ньютона), визначено нижче в часовій області. Крім того, спин ЕРС пропорційна кутовій швидкості двигуна. Закон Фарадея може бути використаний для опису напруги, що створюються зміною магнітного поля в котушці. Нарешті, закон Лоренца, який описує силу на котушці в магнітному полі, приводить до збільшення обертаючого моменту, пропорційного струму двигуна. Ці рівняння показані нижче в часовій області.

застосований ШИМ-сигнал [с]

зворотній ЕРС [В]

*i –* сила струму мотора [А]

*R* – опір катушок і обмоток [Ом]

*L* – індуктивність катушок [Г]

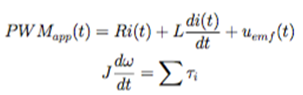
*J* – інерційна нагрузка [кг]

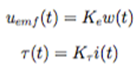
константа ЕРС [В / рад]

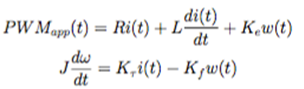
константа арматури [Н / рад]

константа тертя мотора [Н / рад]

*ω* – швидкість мотора [рад / с]

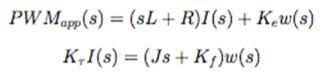


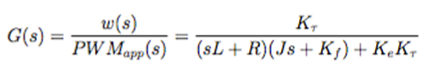


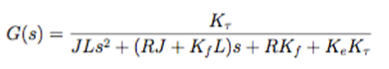


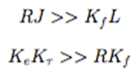
* + - 1. **Ідентифікація динаміки двигуна**

Для того, щоб моделювати динаміку системи двигуна, нам потрібно розрахувати передатню функції, яка описує динаміку двигуна. Вихідна функція передачі - це швидкість гвинта, а вхід - напруга двигуна. Передатня фуекція дозволить нам розрахувати константи часу двигуна та коефіцієнт посилення постійного струму. У рівняннях нижче рівняння перетворюються з часового домену на частотну область, використовуючи перетворення Лапласа [81, 82, 83].



















Кілька припущень зроблено для зменшення передатньої функції для наближення системи першого порядку. Сила внаслідок тертя в безконтактному двигуні постійного струму обумовлена опором і, отже, дуже мала. Крім того, оскільки двигун малий, ми можемо вважати низьку індуктивність і нехтувати цим терміном. Це призведе до передатньої функції вище. Крім того, розраховуються рівняння для постійної часу і DC посилення.

Замість того, щоб намагатися виміряти індивідуальні змінні, на основі розробленого експерименту [78, 79, 80] було проведено вимірювання моторної реакцыъ та оцінки параметрів двигуна. Вхід передатньої функції - це напруга двигуна, а вихід - швидкість валу двигуна. Ми можемо контролювати вхід двигуна, використовуючи ESC, і вимірювати швидкість гвинта за допомогою фотоперетворювача [77].

Arduino Uno використовувався для керування вхідним сигналом двигуна та записом швидкості гвинта. Це популярний мікро-контролер на базі ATmega328Pchipset [78]. Він включає в себе кілька цифрових та аналогових вхідних та вихідних каналів, а також USB-з'єднання. Ідея полягає в тому, щоб відправити ESC різними входами, використовуючи формат PWM, який потім переводиться в напругу двигуна. Фотоперетворювач використовує світлодіод та оптичний приймач для створення простої схеми. Коли об'єкт проходить між світлодіодом та приймачем, електричний ланцюг замикається. Ми можемо використовувати цей датчик для виявлення оберту гвинта при кожному розриві ланцюга, запису мітки часу та розрахунку швидкості гвинта. Ці дані можна потім проаналізувати, використовуючи скрипт MATLAB, щоб оцінити коефіцієнт посилення постійного струму та часовій константі та перевірити модель двигуна.

У цьому блозі [78] повідомляється, як налаштувати схему електроніки. Зверніть увагу, це розмір висувного резистора. Оскільки ми будемо читати вхідний фотоперетворювач на цифровий вхідний канал, цей резистор допомагає витягнути напругу ланцюга на землю, коли ланцюг відкритий, створюючи більш чітке розходження між відкритою та замкненою схемою, записаними як 0 або 1. На малюнку нижче показано повне налаштування електроніки [78, 79, 80].

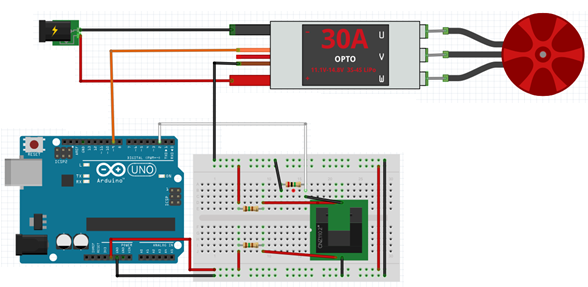


Рис. 29. Схема тестування безщіткового мотору

Хоча ми використовували деякі припущення для спрощення рівнянь руху, фактична система нелінійна через аеродинамічні сили опору на пропелері. Тому модель вище відома як лінійна часова інваріантна система. Оскільки рівняння лінеаризовані, модель буде точною лише навколо вибраного заданого значення. Зазначена точка нахилу вибирається для точки лінеаризації. Тому, коли ми надсилаємо команди двигуна в даному експерименті, ми будемо зберігати значення в околиці вхідного заданого значення.

З попереднього калібрування двигунів знаємо, що мінімальне значення дроселя становить близько 1000 мікросекунд, а максимальне значення дроселя становить близько 2000 мікросекунд. Під час зависання коптера, наш вхід до заданого значення становить близько 1425 мікросекунд. Сценарій Arduino потім відправляє команди двигуна близько 1425 на 5 секунд кожна. Переривання використовується для вимірювання часу, коли пропелер проходить через фотоперетворювач і замикає ланцюг. Виходом цього експерименту є файл .csv зі стовпчиками, що містять мітку часу, вхід PWM та швидкість гвинта.

Потім ми використовуємо сценарій Matlab (Додаток Г) для введення файлу .сsv і аналізу даних. Передатна фукція розроблена у формі, показаної вище, з коефіцієнтом посилення постійного струму та постійної часу, як змінні. Вхід до передатної функції - це набір команд PWM, надісланих на фактичний двигун в експерименті. Виходом передатної функції є відповідний симуляція обертів моменту гвинта. Ці значення накладаються на експериментальні значення для визначення невідомих системних параметрів. Використовуючи константа часу 0.15 секунд і коефіцієнт підсилення 6.5, знаід час польотучення імітації RPM точно відповідають критеріям експерименту, перевіряючи розроблену модель.

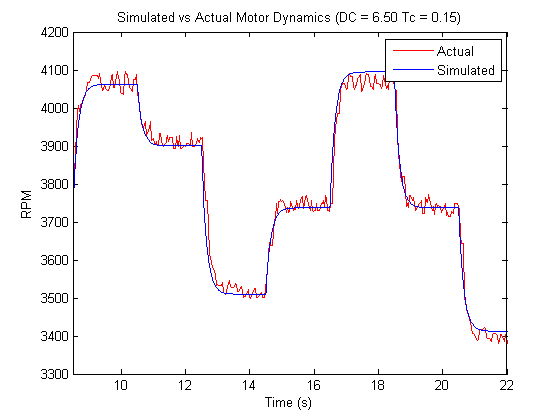
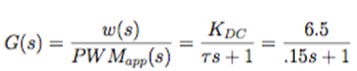
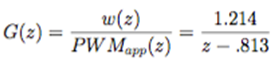


Рис. 30. Результат експерименту порівняння виміряної роботи двигуна та спрощеної моделі.

Для реалізації моделі двигуна, використовуючи фактичні вхідні дані ШІМ, передатна функція повинна бути перетворена з безперервного часу на дискретну модель часу. Це було зроблено за допомогою команди Matlab c2d. Було використано апроксимацію для нульового порядку, а також час вибірки ШІМ 30 Гц, який розраховується із зареєстрованих даних польоту. Нижче наведені розрахунки для перетворення моделі з частотної області на часовий домен, а також для розрахунку остаточного різнорідного рівняння. Зміна перетворення використовується для розрахунку рівняння різниці. Z-конвертація використовується в цифрових системах керування, подібних до того, як перетворення Лапласа було використано раніше в системах безперервного керування. Дане рівняння використовується для оцінки швидкості двигуна в реальному часі, враховуючи поточні значення ШІМ. Оскільки швидкості двигуна є входом до рівнянь руху квадрокоптера, це рівняння дозволяє нам оцінити швидкість двигуна з наших раніше зареєстрованих даних для вдосконалення нашої моделі квадрокоптера. коефіцієнт підсилення постійного струму









 - роторний момент інерції

* + - 1. **Коефіцієнт тяги**

Коефіцієнт тяги можна оцінити за допомогою параметрів двигуна, визначених вище. Нагадаємо, що коефіцієнт тяги - пропорційна величина, яка пов'язує тягу кожного двигуна з кутовою швидкостю пропелера. Це найпростіше для обчислення в точці нахилу, оскільки ми знаємо, що тяга кожного двигуна дорівнює чверті маси квадрокоптера, і ми можемо оцінити швидкість гвинта за раніше зібраними даними. Розрахунок коефіцієнта тяги наведено нижче [83].

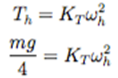
тяга для зависання [Н]

кутова швидкість мотора для зависання [рад/с]

коефіцієнт тяги [кг ]

*m* – маса квадрокоптера [кг]

*g* – сила тяжіння Землі [м /]







* + - 1. **Максимальна швидкість двигуна**

Для підвищення точності моделювання ми можемо встановити обмеження на швидкість двигуна. Це не дозволить системі керування розраховувати сигнали керування, які не є реальними на фактичній системі квадрокоптера. Розміщення лімітів насиченості на швидкості двигуна також допомагає налаштувати контроль вихідних значень та запобігає використанню неефективних керуючих результатів. Команди ШІМ створюють робочий цикл, який може співвідноситись з величиною напруги, що надсилається до двигуна. З наших попередніх експериментів ми знаємо, що квадрук зависає біля команди ШІМ 1425, що відповідає напрузі приблизно 5.35 вольта. Для даного сигналу ШІМ, ми оцінюємо, що гвинт буде крутитися приблизно на 3745 оборотів в хвилину. Від специфікації [56] двигуна, розвантажена постійна (Kv) швидкості двигуна становить 880 об./хв. Наш навантажений Kv становить 700. Дивлячись на дані з двигуна, кожен двигун виробляє 1380г тяги 20A і 210W на повній потужності, яка дорівнює 10.5 вольтам. Тому ми можемо сподіватися, що гвинт не буде крутитися швидше, ніж 7350 оборотів в хвилину.

[кг м]

[кг / ]

[кг c]

[кг c]

[кг c]

* 1. **Розробка системи керування**

Даний розділ присвячено розробці кількох різних стратегій керування. Ці системи керування розробляються на основі моделі квадрокоптера, виведеної в розділі «Розробка і аналіз рівнянь квадроторної системи», реалізованої в середовищі моделювання та підтвердженої на фактичній системі. Системи керування нижнього рівня будуть працювати з високою швидкістю і контролювати орієнтацію квадрокоптера. Ці контролери покладаються на бортові вимірювання ІНС. На більш високому рівні, також можна контролювати поступальне положення квадрокоптера. Залежно від використовуваного датчика, це може бути глобальним або відносним поступальним положенням. Ці датчики можуть бути GPS, камерою, лазером або іншими датчиками. Нижче описано деталі розробки різних систем керування.

* + 1. **Загальна стратегія керування**

Квадрокоптер - це класично неповна система. Квадрокоптер здатний рухатися в 6 ступенів свободи (3 поступальні та 3 обертальні), але має тільки 4 контрольні входи (швидкість кожного мотора). Хоча квадрокоптер може рухатися прямо на вертикальній осі z, не змінюючи жодного іншого стану, він повинен змінити його відношення до переміщення по горизонтальній осі x і y. Оскільки керувати 6 ступенями свободи неможливо лише за допомогою 4 входів керування, ми замість цього розробляємо контролери для стабілізації навколо бажаних позицій x, y, z та бажаного курсу. Квадрук повинен мати можливість безпечно пересуватися в потрібне положення при збереженні стабільних кутів тангажу та крену. Сучасні квадротери мають електронні контролери швидкості (ESC), які контролюють кутові швидкості кожного ротора. Це призводить до наступних входів прямого контролю:

u(1) - результуюча тяга чотирьох роторів,

u(2) - різниця тяги між двигунами на осі x, що призводить до зміни кута тангажу і наступного руху в бічному x напрямку.

u(3) - різниця тяги між двигунами на осі y, що призводить до зміни кута крену та наступного руху в бічному напрямку у.

u(4) - різниця моменту між годинникової та проти годинникової стрілки, що призводить до моменту, який обертає квадрокоптер навколо вертикальної осі z.

Ми можемо використовувати рівняння для чистої тяги та рівнянь моментів для визначення вхідних даних керування в систему квадрокоптера. Контрольні входи визначені нижче. Зверніть увагу, що інвертування цієї матриці дозволяє нам вирішувати потрібні швидкості двигунів вказані значення для чотирьох вхідних елементів керування. Тому, якщо ми можемо обчислити контрольні входи, ми можемо навести ці значення на потрібну швидкість для кожного двигуна. Ці швидкості можуть бути відправлені до ESC.

загальна сила для зависання в звязній СК

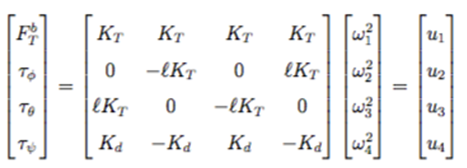
моменти вздовж відповідних осей в звязній СК

загальна тяга від усіх моторів

керуючий вхід по крену

керуючий вхід по тангажу

керуючий вхід по курсу

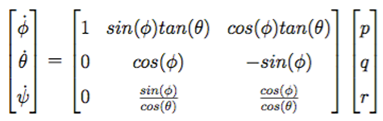


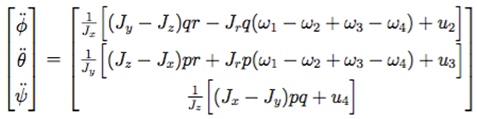
Загальна стратегія керування покладеться на 3 підсистеми керування. Датчики на борту квадрокоптера вимірюють кутові швидкості та поступальне прискорення. Ці значення порівнюються з бажаним значенням орієнтації та положення для розрахунку сигналу помилки, який подають у системи керування. У першій підсистемі помилка позиції - це вхідний сигнал, а бажані кути крену та тангажу є виходами. Далі, помилки тангажу, крену та висоти вводяться в підсистему орієнтації. Ця система виводить бажані кутові швидкості для кожної осі. Вони потрапляють в остаточну систему керування. Виходи цієї системи є вхідними елементами керування, які потім можуть бути віднесені до бажаної швидкості двигуна. Ця загальна стратегія може бути використана для розробки систем керування різного рівня складності.

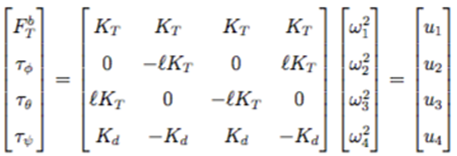
* + 1. **Спрощені рівняння руху**

Складна модель руху квадрокоптера ускладнює отримання точного розуміння поведінки квадрокоптера та усунення несправностей систем керування при моделюванні. Для цього дослідження розглядаються лише основні елементи, що описують поведінку квадрокоптера під час наведення курсу та бажаний експлуатаційний стан. Елементи, які значно впливають на поведінку квадрокоптера на високій швидкості, ігноруються. Ми можемо зробити декілька припущень, які зменшують складність нелінійних рівнянь, описаних вище, для розробки базових стратегій керування. Тоді ми можемо збільшити складність моделі системи для розробки більш точних систем керування у майбутньому. Модель, отримана в розділі «Розробка і аналіз рівнянь квадроторної системи», наведена нижче з введеними контрольними вкладами.

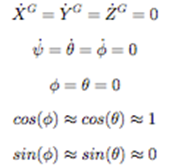


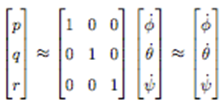


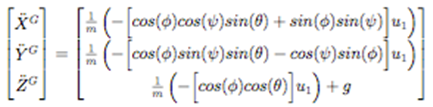




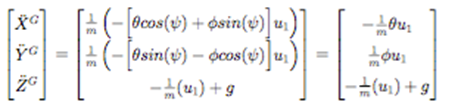
Первинне припущення, яке було зроблено для спрощення моделі, полягає в тому, що квадрокоптер буде працювати навколо стабільного курсу з невеликими кутами орієнтації та мінімальними обертальними та поступальними швидкостями і прискореннями. Оскільки немає аеродинамічних піднімальних поверхонь, припускаємо, що аеродинамічні сили та моменти є незначними. Ефекти, які вважаються незначними, розглядаються як порушення в системі керування та можуть бути компенсовані відповідним дизайном системи керування. Ці припущення описані математично, а також отримані спрощені рівняння руху. Наведені нижче рівняння продовжують зменшуватися у складності, оскільки робляться інші припущення.

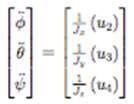












Хоча ці рівняння спрощують процес проектування та аналізу керування, ми все ще можемо використовувати більш точні нелінійні рівняння руху в середовищі симуляції для оцінки надійності нашої системи керування. Зверніть увагу, що в остаточних рівняннях поступальний рух було відокремлено від орієнтації, припускаючи, що курс залишається близько 0 градусів. Рух уздовж глобальної осі x і y можна контролювати, самостійно змінюючи значення крену та рискання. Крім того, для того, щоб квадрокоптер підтримував стабільну висоту, вектор тяги, поділений на масу квадрокоптер, повинен бути рівним прискоренню, викликаному силою тяжіння. Припущення, необхідні для лінеалізації моделі, зроблені, тому модель може бути використана як основа для проектування системи керування.

* + 1. **Спрощений ПІД-контролер**

Найпоширенішим типом системи керування є система керування пропорційно-інтегрально-похідною (ПІД). ПІД-регулятор називається системою зворотного зв'язку замкнутого циклу. Контролер обчислює різницю між фактичним та бажаним станом і виробляє значення помилки. Вимірювання з датчиків квадрокоптера відновлюються, щоб обчислити цей сигнал помилки, закриваючи цикл від виходу до вводу. Цей контролер користується популярністю, оскільки він трохи інтуїтивно налаштовується, легко реалізується в коді з малим обсягом обробки, а також використовує вимірювання, доступні більшості систем на борту. Вихід будь-якої системи керування ПІД-контролем - це кероване значення u, яке приведе систему до потрібного стану. Дизайн ПІД-контролера показана нижче як у часовому домені, так і в частотному домені.

*u* – керуючий вхід

*P* – пропорційна складова

*I* – інтегральна складова

*D* – диференційна складова

коефіцієнт підсилення пропорційної складової

коефіцієнт підсилення інтегральної складової

коефіцієнт підсилення диференційної складової

*t* – миттєва швидкість

бажане значення стану *х* в момент часу *t*

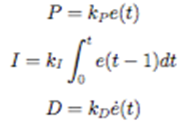
виміряне значення стану *х* в момент часу *t*

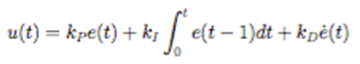
помилка значення в момент часу *t*

похідна від помилки в момент часу *t*











Пропорційний регулятор - це будь-яке число більше ніж 0, і в результаті виникає значення, яке лише в декілька разів кратна сигналу помилки. Інтегральний регулятор є кратним суми похибки, накопиченої в часі. Цей регулятор реагує на накопичення будь-якої залишкової стаціонарної помилки. Ця помилка часто являє собою зберюючі впливи (вітер, віддалений простір, помилки моделювання тощо), які не можна було враховувати за пропорційними чи похідними умовами. Тому інтегральний термін використовується для здійснення невеликих, остаточних коригувань для зменшення остаточної стаціонарної помилки. Похідний регулятор кратний швидкості зміни помилки. Цей регулятор гальмує реакцію системи керування і використовується для зменшення перевиконання системи над бажаним станом. Представлення структурної схеми основного ПІД-регулятора показано нижче.

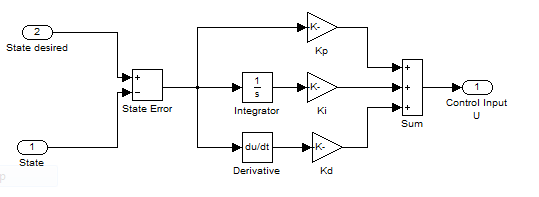


Рис. 31. Структурна схема основного ПІД-регулятора

Існують різні методи, які можуть бути використані для налаштування ПІД-контролера та визначення найкращих значень керуючих впливів за заданими параметрами відповідності системи. Для цього проекту ці значення були визначені в симуляції для забезпечення нідійного переміщення. Зверніть увагу, що для ручного RC польоту квадрокоптера тільки контролер орієнтації є необхідним. Контролер позиції потрібен лише при використанні зовнішнього датчика локалізації, такого як навігація GPS та провідної точки. Також, є деякі характеристики фізичної системи, які основний ПІД-контролер може не точно визначити. До них відносяться затримки вимірювань датчиків та реакції двигуна, а також інерція транспортного засобу.

* + 1. **Розробка системи керування**

Контроль квадрокоптера часто реалізується за допомогою циклів керування. Внутрішній ланцюг (контор керування) контролює кутові швидкості кожної осі квадрокоптера. Даний ланцюг повинний працювати на високій частоті через швидку динаміку квадрокоптера. Наступний ланцюг контролю висоти та орієнтації забезпечує орієнтацію та висоту квадрокоптера. Малі зміни орієнтації безпосередньо пов'язані з поступальним прискоренням. Це означає, що дані помилки можуть призвести до великих та небажаних переміщень у процесі руху. Однією з переваг для керування циклом контролю швидкості на високій частоті є те, що бортові датчики, такі як акселерометр і гіроскоп, також працюють на високій частоті. Це означає, що ми можемо скористатися найновішими та точнішими вимірами при обчисленні контрольних значень. Даний контролер висоти / орієнтації може отримати бажану висоту та орієнтацію від пульта дистанційного керування або зовнішнього ланцюга контролю.

Зовнішній ланцюг керує поступальним рухом уздовж осі x та y. Цей контролер виводить бажані кути орієнтації для внутрішнього контролера на основі помилки на осі x і y. Для цього зовнішній контролер має декілька входів. Це можуть бути точки шляху, створені наземною станцією керування, алгоритмом планування шляху або, наприклад, візуальною системою. В Додатку Ґ наведено блок-схему, яка демонструє взаємозв’язок контролерів із системою керування квадрокоптера.

* + - 1. **Контур керування позиції квадрокоптера в глобальній СК**

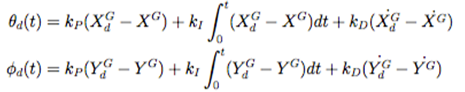
Тоді як регулятор орієнтації / висоти може підтримувати стабільний курс для квадрокоптера та відстежувати бажані вхідні значення орієнтації / висоти з RC або GCS, він не може контролювати поступальний рух квадрокоптера. Незалежно від того, наскільки точний може бути контролер орієнтації, маленькі кути орієнтації призведуть до поступального руху або дрейфу квадрокоптера. Для запобігання такого поступального дрейфу позиції здійснюється контролер позиції. Для того, щоб цей контролер працював, потрібно виконувати певну відносну або глобальну локалізацію. Як правило, це робиться за допомогою камери для позиціонування дрона (візуальне спостереження) або GPS для глобального позиціонування дрона. Даний зовнішній контур може працювати на повільній швидкості в порівнянні з контролером внутрішнього контору керування. Входи до контролера позиції - це вимірювання положення, а виходи - бажані кути тангажу і крену, щоб маневрувати квадрокоптер до потрібного положення x і y. Цей контролер також забезпечує слідування по траєкторії, коли рядок послідовних точок траєкторії може бути поданий на контролер позиції, щоб квадрокоптер дотримувався певного тривимірного шляху.

бажане глобальне положення X [м]

бажане глобальне положення Y [м]

бажаний кут крену [рад]

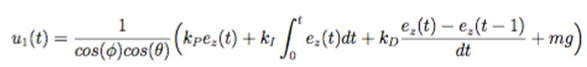
бажаний кут тангажу [рад]

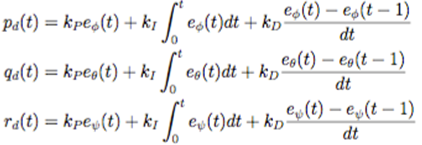


* + - 1. **Контур керування орієнтації та висоти**

У даному розділі детально описано розробка системи керування орієнтації та висотою. Спочатку ми хочемо реалізувати цей контролер для стабілізації квадрокоптера навколо стабільного наведення. Це означає, що всі бажані кути дорівнюють 0, і ми встановлюємо бажану висоту до деяких низьких значень для цілей налаштування. Пізніше ці бажані значення та значення висоти можуть змінюватися залежно від вхідних значень від RC або зовнішнього контуру керування. Оскільки наша спрощена модель видаляє будь-яку взаємну зв'язність в рівняннях руху, керування відношенням може бути реалізовано за допомогою незалежного вхідного керування для кожного кута Ейлера.

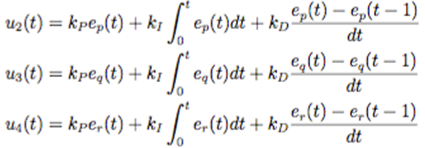
Контроль висоти аналогічний контролю загальної тяги квадрокоптера. Це важливо, оскільки цей контрольний вхід також впливає на рух уздовж осі x та y. Це видно в поступального рівняннях руху, де *u1* - вхід у напрямках *x* і *y*. Тому більш високі значення тяги призводять до більш агресивних рухів в *х* і *у* напрямках, заданих постійними кутами тангажу та крену. У нашій спрощеній моделі вертикальне прискорення відокремлено від кута нахилу. Однак це не відбувається, коли квадрокоптер обертається, оскільки компоненти вертикального прискорення також знаходяться в площинах *х* і *у*. Тому контроль висоти компенсує як зміщення гравітації, так і відхилення узагальненої тяги за рахунок нахилу. Повний ПІД-контролер докладно описані нижче.





* + - 1. **Контур керування кутовою швидкістю**

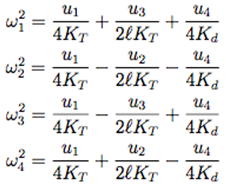
Контролер кутової швидкісті - це контролер найнижчого рівня, розроблений для системи квадрокоптерів. Даний контролер отримує бажані кутові швидкості від регулятора орієнтації. Потім він обчислює помилку між бажаними показниками та швидкістю, виміряною гіроскопом. Дана помилка потім використовується для обчислення трьох контрольних входів моменту. Вони поєднуються з входом керування тягою, обчисленого контролером висоти, які потім перетворюються на потрібну швидкість двигуна і направляється на мотори через регулятори обертів ESC.



* + - 1. **Керування двигуном**

З урахуванням входів керування, можна обчислити індивідуальні швидкості обертів, необхідні для кожного двигуна. Ці швидкості будуть відправлені до ESC, після чого вони будуть надсилати команди до приєднаного двигуна, щоб вони оберталися на бажану швидкість для отримання бажаної тяги та моментів. Бажані швидкості двигуна можна обчислити, беручи зворотню керуючу матрицю, розроблену в розділі вище, як показано нижче. Зверніть увагу, що всі двигуни містять деяку складову вхідних елементів керування тягою та поворотом з відповідними входами для кожної осі. Двигуни, що контролюють момент крену, включають в себе компоненти вхідного регулювання крену, а двигуни, що контролюють момент тангажу, включають компоненти керуючого входу тангажу. Ці рівняння призначені для "+" квадроторної орієнтації і повинні бути змінені, якщо квадрокоптер повинен був в "X" конфігурації.





Важливо також встановити обмеження на керуючі входи, які можна обчислити системою керування. Ці керуючі входи перетворюються на швидкості двигуна, які потім надсилаються до ESC і, згодом, до двигунів. Тому ми не хочемо використовувати керуючі входи, які призводить до бажаної швидкості двигуна, що не є можливим для реальної системи. Обчислення обмежень контролю наведено нижче на основі фізичної системи.

*ω* – кутова швидкість мотора [рад/с]

максимальна швидкість мотора [920 рад/с]

мінімальна швидкість мотора [0 рад/с]

[кг / ]

[кг м]

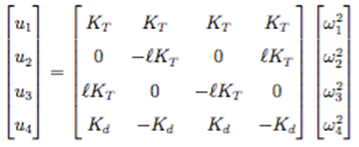
*l =* 0.56 [м]

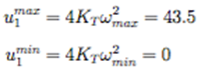
загальна тяга від усіх моторів [кг / ]

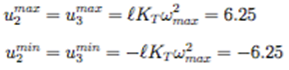
керуючий вхід по крену [кг / ]

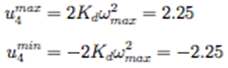
керуючий вхід по тангажу [кг / ]

керуючий вхід по курсу [кг / ]









* + - 1. **Основні модифікації ПІД – «Похідний удар»**

У цьому розділі описано кілька модифікацій, які можна внести до основного ПІД-регулятора, описаного вище, з метою поліпшення польоту та стабільності. Бретт Бейергард має більш детальні пояснення цих модифікацій на своєму веб-сайті [84].

Перша модифікація мінімізує так званий "похідний удар". Похибка розраховується як різниця між заданою і поточною вхідними вимірами. Отже, похідна від помилки дорівнює різниці між похідною від заданого значення та похідною від вимірюваного вхідного значення. Це означає, що будь-яка зміна заданого значення призводить до значного збільшення її похідної. Для більшості додатків задане значення залишається відносно постійним. При виконанні бажаної траєкторії задане значення може продовжувати змінюватися. Незважаючи на це, ці миттєві зміни у заданому значенні призведуть до великих похідних, які будуть подані в ПІД-контролер. Враховуючи високу нелінійну динаміку квадрокоптера, це може призвести до нестабільності. Замість цього ми припускаємо, що похідна від заданого значення часто не змінюється і встановлюється її похідна на 0. Це призводить до похідної помилки, яка дорівнює негативній похідной від вимірюваного вхідного сигналу. Похідне значення часто можна виміряти за допомогою бортового датчика. Модифікація призводить до наступної реалізації блок-схеми. Зверніть увагу, що значення Kd, коефіцієнт похідного регулятора, має протилежний знак Kd, що використовується в традиційній реалізації ПІД.

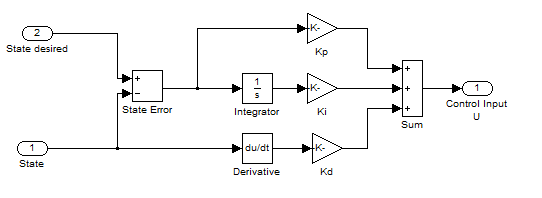


Рис. 32. Структурна схема модифікації ПІД-регулятора

* + - 1. **Основні модифікації ПІД - Anti-WindUp**

Друга модифікація покращує стійкість, встановлюючи обмеження на інтегральний регулятор керування. Як обговорювалося раніше, інтегральний член обчислює суму помилки з часом. Метою цього регулятора є компенсація будь-яких постійних порушень, таких як навколишнє середовище або помилки моделювання. Цей термін найбільше впливає, коли система вже досягла стійкого стану, але має постійну помилку. Інтегральний термін збільшить контрольну потужність, щоб повільно зменшувати помилку стаціонарного стану. Проте, якщо є велика помилка через порушення або раптова зміна заданого значення, інтегральний регулятор може швидко стати великим значенням, оскільки воно продовжує додаватися в той час, як система наближається до бажаного заданого значення. Навіть коли система наближається до потрібного значення, інтегральний термін залишається дуже великим і може змусити систему перевищити, стати нестабільною або коливатися. Крім того, завдяки великому інтегральному нарощуванню, якщо задане значення змінюється в протилежному напрямку, інтегральний термін затримає відповідь контролера, оскільки він очікує зменшення інтегрального значення. Щоб це виправити, ми вмикаємо інтегральний регулятор лише тоді, коли ми перебуваємо в області навколо стабільного стану, який контролюється. Це дозволяє стабільним чином усунути будь-яку стаціонарну помилку. Також ми можемо наситити вихід контролера, щоб відповідати мінімальним і максимальним значенням входу мотора.

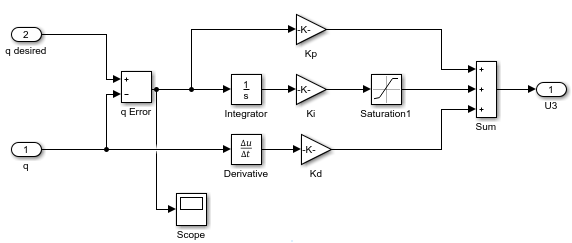


Рис. 33. Структурна схема модифікації ПІД-регулятора з обмеженням по інтегралу

* 1. **Симуляція динаміки польоту квадрокоптера в програмному забезпечені Matlab**

На основі вище описаного матеріалу з розділів 5.1-5.3 було розроблено програмне забезпечення засобами Matlab для моделювання динаміки польоту квадрокоптера. Структурна схема системи керування засобами simulink Matlab зображено в Додатку Ґ.

В даному розділі описуються деталі розробки надійного та точного середовища моделювання. Точне моделювання поєднує в собі рівняння руху, отримані в розділі 5.1, конкретні значення параметрів, які оцінюються в розділі 5.2, та контролера, розробленого в розділі 5.3. Всі ці частини необхідні для створення точного середовища моделювання. Точне середовище моделювання дозволяє операторам проектувати та перевіряти системи керування, фільтри, алгоритми планування шляху, перш ніж вони будуть реалізовані на фізичній системі в реальному світі. У наступних розділах описано код, необхідний для реалізації симуляції руху квадрокоптера в Matlab. Де це доречно, конкретний код або блоки виділяються для більш докладного опису.

Нижче наведено короткий опис основних файлів і функцій, необхідних для запуска та імітації керування квадрокоптера.

Симулятор запускається за допомогою файла quadrotor\_sim.m. Цей файл ініціалізує середовище моделювання за допомогою наступних команд. Створено глобальну змінну Quad, яка буде містити всі перемінні квадрокоптера.

% Add Paths

addpath additional

%% Initialize Workspace

clear all;

close all;

clc;

global Quad;

%% Initialize the plot

init\_plot;

plot\_quad\_model;

%% Initialize Variables

quad\_variables;

quad\_dynamics\_nonlinear;

Функція init\_plot.m створює тривимірне середовище, буде візуалізовано рух квадрокоптера.

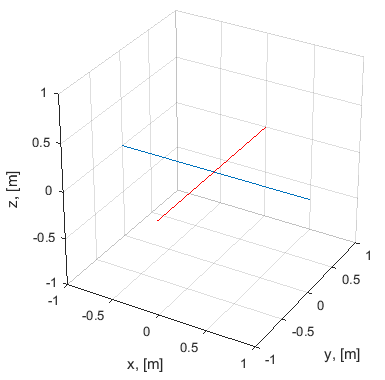


Рис. 34. Тривимірне середовище

Далі визивається plot\_quad\_model.m. Цей сценарій починається з завантаження файлу Quad\_plotting\_model.mat, який створюється запуском define\_quad\_model.m. Цей файл .mat використовує фізичні розміри квадрокоптера, такі як довжина і товщина плеча рами, радіус гвинта, щоб визначати вершини кожного плеча та двигуна в трьохвимірному середовищі для цілей побудови. Сценарій plot\_quad\_model.m дає початкову модель квадрокоптера в середовищі побудови.

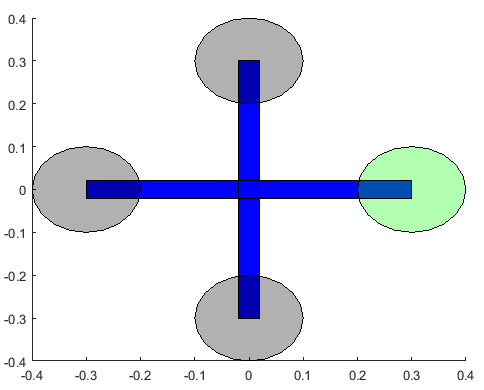


Рис. 35. Візуальна модель квадрокоптера

Далі запускається quad\_variables.m, який визначає основні змінні для симуляції, включаючи фізичні параметри квадрокоптера, початкові та бажані умови, параметри моделювання та коефіцієнтів контролера. Значення фізичних параметрів взято з розділу 5.2.

Зауважимо, що в алгоритм симуляції закладена частота обробки головного циклу в 100 Гц. Дане число було взято з фізичної реальної моделі APM2.6.

Функція quad\_dynamics\_nonlinear.m використовує початкові значення для розрахунку початкових швидкостей і прискорень моделі квадрокоптера. У цих моделях використовуються нелінійні рівняння руху, отримані в розділі 5.1. Код, що представляє ці рівняння, наведено нижче. Зверніть увагу, що дискретна інтеграція використовується для обчислення швидкостей та позицій зі значень прискорення.

function quad\_dynamics\_nonlinear

global Quad;

%% Update Accelerations

Quad.X\_ddot = (-[cos(Quad.phi)\*sin(Quad.theta)\*cos(Quad.psi)+sin(Quad.phi)\*sin(Quad.psi)]\*Quad.U1-Quad.Kdx\*Quad.X\_dot)/Quad.m;

Quad.Y\_ddot = (-[cos(Quad.phi)\*sin(Quad.psi)\*sin(Quad.theta)-cos(Quad.psi)\*sin(Quad.phi)]\*Quad.U1-Quad.Kdy\*Quad.Y\_dot)/Quad.m;

Quad.Z\_ddot = (-[cos(Quad.phi)\*cos(Quad.theta)]\*Quad.U1-Quad.Kdz\*Quad.Z\_dot)/Quad.m+Quad.g;

Quad.p\_dot = (Quad.q\*Quad.r\*(Quad.Jy - Quad.Jz) - Quad.Jp\*Quad.p\*Quad.Obar + Quad.l\*Quad.U2)/Quad.Jx;

Quad.q\_dot = (Quad.p\*Quad.r\*(Quad.Jz - Quad.Jx) + Quad.Jp\*Quad.q\*Quad.Obar + Quad.l\*Quad.U3)/Quad.Jy;

Quad.r\_dot = (Quad.p\*Quad.q\*(Quad.Jx - Quad.Jy) + Quad.U4)/Quad.Jz;

Quad.phi\_dot = Quad.p + sin(Quad.phi)\*tan(Quad.theta)\*Quad.q + cos(Quad.phi)\*tan(Quad.theta)\*Quad.r;

Quad.theta\_dot = cos(Quad.phi)\*Quad.q - sin(Quad.phi)\*Quad.r;

Quad.psi\_dot = sin(Quad.phi)/cos(Quad.theta)\*Quad.q + cos(Quad.phi)/cos(Quad.theta)\*Quad.r;

Наступне в quadrotor\_sim.m - це цикл, який запускає фактичне моделювання, коли вся ініціалізація завершена. Релевантний код показаний нижче.

Quad.X\_prev = Quad.X;

Quad.Y\_prev = Quad.Y;

Quad.Z\_prev = Quad.Z;

%calcu < 2000

%% Run The Simulation Loop

while Quad.t\_plot(Quad.counter-1)< max(Quad.t\_plot) && landing ~= 1

% Measure Parameters (for simulating sensor errors)

sensor\_meas;

% Filter Measurements

% Kalman\_phi2;

% Kalman\_theta2;

% Kalman\_psi2;

% Kalman\_Z2;

% Kalman\_X2;

% Kalman\_Y2;

%%

% Implement Controller

position\_PID;

attitude\_PID;

rate\_PID;

% Calculate Desired Motor Speeds

quad\_motor\_speed;

% Update Position With The Equations of Motion

quad\_dynamics\_nonlinear;

% Plot the Quadrotor's Position

if(mod(Quad.counter,5)==0)

plot\_quad

% Plot the Quadrotor's track

line([Quad.X Quad.X\_prev],[-Quad.Y -Quad.Y\_prev],[-Quad.Z+Quad.Correction\_axe\_Z -Quad.Z\_prev+Quad.Correction\_axe\_Z],'color','r')

line([Quad.X Quad.X\_prev],[-Quad.Y -Quad.Y\_prev],[0 0],'color','b')

Quad.X\_prev = Quad.X;

Quad.Y\_prev = Quad.Y;

Quad.Z\_prev = Quad.Z;

% campos([A.X+2 A.Y+2 A.Z+2])

% camtarget([A.X A.Y A.Z])

% camroll(0);

Quad.counter;

drawnow

end

Quad.init = 1; %Ends initialization after first simulation iteration

end

Цикл працює протягом часу, заданого t\_plot або поки дрон не здійснить посадку. Спершу виконуються функції імітації шуму датчиків, а також реалізації фільтрів, такого як EKF, для фільтрації вимірювань датчика перед відправленням їх до контролера. В даний час реалізується тільки симуляційний шум датчика. Це зроблено в файлі sensor\_meas.m. Цей файл оновлює глобальну позицію, лінійне прискорення та змінні швидкості обертання, використовуючи модель датчика, виведену в розділі 5.2. Специфічна функція докладно описана нижче.

function sensor\_meas

global Quad;

%% GPS Measurements

if(mod(Quad.counter,Quad.GPS\_freq) == 0)

Quad.X = Quad.X + randn(1)\*Quad.X\_error;

Quad.Y = Quad.Y + randn(1)\*Quad.Y\_error;

%% Barometer Measurements

Quad.Z = Quad.Z + randn(1)\*Quad.Z\_error;

end

%% IMU Measurements

Quad.X\_ddot = Quad.X\_ddot + Quad.x\_acc\_bias + Quad.x\_acc\_sd\*randn(1);

Quad.Y\_ddot = Quad.Y\_ddot + Quad.y\_acc\_bias + Quad.y\_acc\_sd\*randn(1);

Quad.Z\_ddot = Quad.Z\_ddot + Quad.z\_acc\_bias + Quad.z\_acc\_sd\*randn(1);

Quad.p = Quad.p + Quad.x\_gyro\_bias + Quad.x\_gyro\_sd\*randn(1);

Quad.q = Quad.q + Quad.y\_gyro\_bias + Quad.y\_gyro\_sd\*randn(1);

Quad.r = Quad.r + Quad.z\_gyro\_bias + Quad.z\_gyro\_sd\*randn(1);

%% Plotting Variables

Quad.X\_ddot\_meas\_plot(Quad.counter) = Quad.X\_ddot;

Quad.Y\_ddot\_meas\_plot(Quad.counter) = Quad.Y\_ddot;

Quad.Z\_ddot\_meas\_plot(Quad.counter) = Quad.Z\_ddot;

end

Потім визивається функція position\_PID.m, яка діє як контролер позиції. Виходи цієї функції є бажані кути крену та тангажу, є входами до attitude\_PID.m, який є контролером орієнтації / висоти. Обидва є ПІД-регуляторами та реалізують рівняння, отримані в розділі 5.3. Код поступального ПІД-регулятора показаний нижче для довідки. Зверніть увагу, що бажані позиції в глобальній СК повинні бути перетворені в СК тіла.

function position\_PID

persistent x\_error\_sum;

persistent y\_error\_sum;

global Quad

% initialize persistent variables at beginning of simulation

if Quad.init==0

x\_error\_sum = 0;

y\_error\_sum = 0;

end

%% High Level Position Controller

x = Quad.X;

y = Quad.Y;

z = Quad.Z;

phi = Quad.phi;

theta = Quad.theta;

psi = Quad.psi;

% Rotate Desired Position from GF to BF (Z axis rotation only)

[Quad.X\_des,Quad.Y\_des,Quad.Z\_des] = rotateGFtoBF(Quad.X\_des\_GF,Quad.Y\_des\_GF,Quad.Z\_des\_GF,0\*phi,0\*theta,psi);

% Rotate Current Position from GF to BF

[Quad.X\_BF,Quad.Y\_BF,Quad.Z\_BF] = rotateGFtoBF(x,y,z,phi,theta,psi);

% Rotate Current Velocity from GF to BF

[Quad.X\_BF\_dot,Quad.Y\_BF\_dot,Quad.Z\_BF\_dot] = rotateGFtoBF(Quad.X\_dot,Quad.Y\_dot,Quad.Z\_dot,phi,theta,psi);

% X Position PID controller

x\_error = Quad.X\_des - Quad.X\_BF;

if(abs(x\_error) < Quad.X\_KI\_lim)

x\_error\_sum = x\_error\_sum + x\_error;

end

cp = Quad.X\_KP\*x\_error; %Proportional term

ci = Quad.X\_KI\*Quad.Ts\*x\_error\_sum;

ci = min(Quad.theta\_max, max(-Quad.theta\_max, ci)); %Saturate ci

cd = Quad.X\_KD\*Quad.X\_BF\_dot; %Derivative term

Quad.theta\_des = - (cp + ci + cd); %Theta and X inversely related

Quad.theta\_des = min(Quad.theta\_max, max(-Quad.theta\_max, Quad.theta\_des));

% Y Position PID controller

y\_error = Quad.Y\_des - Quad.Y\_BF;

if(abs(y\_error) < Quad.Y\_KI\_lim)

y\_error\_sum = y\_error\_sum + y\_error;

end

cp = Quad.Y\_KP\*y\_error; %Proportional term

ci = Quad.Y\_KI\*Quad.Ts\*y\_error\_sum;

ci = min(Quad.phi\_max, max(-Quad.phi\_max, ci)); %Saturate ci

cd = Quad.Y\_KD\*Quad.Y\_BF\_dot; %Derivative term

Quad.phi\_des = cp + ci + cd;

Quad.phi\_des = min(Quad.phi\_max, max(-Quad.phi\_max, Quad.phi\_des));

end

Код контролера орієнтації / висоти відображається нижче. Виходи цього контролера - це три бажані кутові швидкості та бажана тяга.

function attitude\_PID

persistent z\_error\_sum;

persistent phi\_error\_sum;

persistent theta\_error\_sum;

persistent psi\_error\_sum;

global Quad

% initialize persistent variables at beginning of simulation

if Quad.init==0

z\_error\_sum = 0;

phi\_error\_sum = 0;

theta\_error\_sum = 0;

psi\_error\_sum = 0;

end

% % Measurement Model

% if(Quad.ground\_truth)

% phi = Quad.phi;

% theta = Quad.theta;

% psi = Quad.psi;

% end

%

% if(Quad.sensor\_unfiltered)

% phi = Quad.phi\_meas;

% theta = Quad.theta\_meas;

% psi = Quad.psi\_meas;

% end

%

% if(Quad.sensor\_kf)

% phi = Quad.phi;

% theta = Quad.theta;

% psi = Quad.psi;

% end

phi = Quad.phi;

theta = Quad.theta;

psi = Quad.psi;

%% Z Position PID Controller/Altitude Controller

z\_error = Quad.Z\_des\_GF-Quad.Z\_BF;

if(abs(z\_error) < Quad.Z\_KI\_lim)

z\_error\_sum = z\_error\_sum + z\_error;

end

cp = Quad.Z\_KP\*z\_error; %Proportional term

ci = Quad.Z\_KI\*Quad.Ts\*z\_error\_sum; %Integral term

ci = min(Quad.U1\_max, max(Quad.U1\_min, ci)); %Saturate ci

cd = Quad.Z\_KD\*Quad.Z\_dot; %Derivative term

Quad.U1 = -(cp + ci + cd)/(cos(theta)\*cos(phi)) + (Quad.m \* Quad.g)/(cos(theta)\*cos(phi)); %Negative since Thurst and Z inversely related

Quad.U1 = min(Quad.U1\_max, max(Quad.U1\_min, Quad.U1));

%% Attitude Controller

% Roll PID Controller

phi\_error = Quad.phi\_des - phi;

if(abs(phi\_error) < Quad.phi\_KI\_lim)

phi\_error\_sum = phi\_error\_sum + phi\_error;

end

cp = Quad.phi\_KP\*phi\_error;

ci = Quad.phi\_KI\*Quad.Ts\*phi\_error\_sum;

ci = min(Quad.p\_max, max(-Quad.p\_max, ci));

cd = Quad.phi\_KD\*Quad.p;

Quad.p\_des = cp + ci + cd;

Quad.p\_des = min(Quad.p\_max, max(-Quad.p\_max, Quad.p\_des));

% Pitch PID Controller

theta\_error = Quad.theta\_des - theta;

if(abs(theta\_error) < Quad.theta\_KI\_lim)

theta\_error\_sum = theta\_error\_sum + theta\_error;

end

cp = Quad.theta\_KP\*theta\_error;

ci = Quad.theta\_KI\*Quad.Ts\*theta\_error\_sum;

ci = min(Quad.q\_max, max(-Quad.q\_max, ci));

cd = Quad.theta\_KD\*Quad.q;

Quad.q\_des = cp + ci + cd;

Quad.q\_des = min(Quad.q\_max, max(-Quad.q\_max, Quad.q\_des));

% Yaw PID Controller

psi\_error = Quad.psi\_des - psi;

if(abs(psi\_error) < Quad.psi\_KI\_lim)

psi\_error\_sum = psi\_error\_sum + psi\_error;

end

cp = Quad.psi\_KP\*psi\_error;

ci = Quad.psi\_KI\*Quad.Ts\*psi\_error\_sum;

ci = min(Quad.r\_max, max(-Quad.r\_max, ci));

cd = Quad.psi\_KD\*Quad.r;

Quad.r\_des = cp + ci + cd;

Quad.r\_des = min(Quad.r\_max, max(-Quad.r\_max, Quad.r\_des));

end

Нарешті, контролер швидкості здійснюється в функції rate\_PID.m. Цей контур керування приймає бажані кутові швидкості, обчислені з контролера орієнтації, та обчислює три керуючих сигналів. Потім вони накладаються на команди двигуна у наступному коді.

function rate\_PID

persistent p\_error\_sum;

persistent q\_error\_sum;

persistent r\_error\_sum;

global Quad

% initialize persistent variables at beginning of simulation

if Quad.init==0

p\_error\_sum = 0;

q\_error\_sum = 0;

r\_error\_sum = 0;

end

p = Quad.p;

q = Quad.q;

r = Quad.r;

%% Angular Rate Controller

% Roll PID Controller

p\_error = Quad.p\_des - p;

if(abs(p\_error) < Quad.p\_KI\_lim)

p\_error\_sum = p\_error\_sum + p\_error;

end

cp = Quad.p\_KP\*p\_error;

ci = Quad.p\_KI\*Quad.Ts\*p\_error\_sum;

ci = min(Quad.U2\_max, max(Quad.U2\_min, ci));

cd = Quad.p\_KD\*Quad.p\_dot;

Quad.U2 = cp + ci + cd;

Quad.U2 = min(Quad.U2\_max, max(Quad.U2\_min, Quad.U2));

% Pitch PID Controller

q\_error = Quad.q\_des - q;

if(abs(q\_error) < Quad.q\_KI\_lim)

q\_error\_sum = q\_error\_sum + q\_error;

end

cp = Quad.q\_KP\*q\_error;

ci = Quad.q\_KI\*Quad.Ts\*q\_error\_sum;

ci = min(Quad.U3\_max, max(Quad.U3\_min, ci));

cd = Quad.q\_KD\*Quad.q\_dot;

Quad.U3 = cp + ci + cd;

Quad.U3 = min(Quad.U3\_max, max(Quad.U3\_min, Quad.U3));

% Yaw PID Controller

r\_error = Quad.r\_des - r;

if(abs(r\_error) < Quad.r\_KI\_lim)

r\_error\_sum = r\_error\_sum + r\_error;

end

cp = Quad.r\_KP\*r\_error;

ci = Quad.r\_KI\*Quad.Ts\*r\_error\_sum;

ci = min(Quad.U4\_max, max(Quad.U4\_min, ci));

cd = Quad.r\_KD\*Quad.r\_dot;

Quad.U4 = cp + ci + cd;

Quad.U4 = min(Quad.U4\_max, max(Quad.U4\_min, Quad.U4));

end

Далі функція quad\_motor\_speed.m використовується для перетворення керуючих входів на швидкості двигунів, насичення швидкості двигунів на основі характеристик двигуна, а потім перевизначення вхідних керуючих для рівнянь руху. Відповідний код обчислює вхідний сигнал керування для формул зіставлення швидкості двигуна, виведений у розділі Дизайн контролера.

function quad\_motor\_speed

global Quad

% Calculate motor speeds (rad/s)^2

w1 = Quad.U1/(4\*Quad.KT) + Quad.U3/(2\*Quad.KT\*Quad.l) + Quad.U4/(4\*Quad.Kd);

w2 = Quad.U1/(4\*Quad.KT) - Quad.U2/(2\*Quad.KT\*Quad.l) - Quad.U4/(4\*Quad.Kd);

w3 = Quad.U1/(4\*Quad.KT) - Quad.U3/(2\*Quad.KT\*Quad.l) + Quad.U4/(4\*Quad.Kd);

w4 = Quad.U1/(4\*Quad.KT) + Quad.U2/(2\*Quad.KT\*Quad.l) - Quad.U4/(4\*Quad.Kd);

% Apply realistic motor speed limits

if w1 > Quad.max\_motor\_speed^2

w1 = Quad.max\_motor\_speed^2;

end

if w1 < Quad.min\_motor\_speed^2

w1 = Quad.min\_motor\_speed^2;

end

if w2 > Quad.max\_motor\_speed^2

w2 = Quad.max\_motor\_speed^2;

end

if w2 < Quad.min\_motor\_speed^2

w2 = Quad.min\_motor\_speed^2;

end

if w3 > Quad.max\_motor\_speed^2

w3 = Quad.max\_motor\_speed^2;

end

if w3 < Quad.min\_motor\_speed^2

w3 = Quad.min\_motor\_speed^2;

end

if w4 > Quad.max\_motor\_speed^2

w4 = Quad.max\_motor\_speed^2;

end

if w4 < Quad.min\_motor\_speed^2

w4 = Quad.min\_motor\_speed^2;

end

Quad.O1 = sqrt(w1); % Front M

Quad.O2 = sqrt(w2); % Right M

Quad.O3 = sqrt(w3); % Rear M

Quad.O4 = sqrt(w4); % Left M

Quad.O1\_plot(Quad.counter) = Quad.O1;

Quad.O2\_plot(Quad.counter) = Quad.O2;

Quad.O3\_plot(Quad.counter) = Quad.O3;

Quad.O4\_plot(Quad.counter) = Quad.O4;

%% Re-compute traditional control inputs

Quad.U1 = Quad.KT\*(Quad.O1^2 + Quad.O2^2 + Quad.O3^2 + Quad.O4^2);

Quad.U1\_plot(Quad.counter) = Quad.U1;

Quad.U2 = Quad.KT\*Quad.l\*(Quad.O4^2 - Quad.O2^2);

Quad.U2\_plot(Quad.counter) = Quad.U2;

Quad.U3 = Quad.KT\*Quad.l\*(Quad.O1^2 - Quad.O3^2);

Quad.U3\_plot(Quad.counter) = Quad.U3;

Quad.U4 = Quad.Kd\*(Quad.O1^2 + Quad.O3^2 - Quad.O2^2 - Quad.O4^2);

Quad.U4\_plot(Quad.counter) = Quad.U4;

Quad.O = (Quad.O1 - Quad.O2 + Quad.O3 - Quad.O4);

Quad.O\_plot(Quad.counter) = Quad.O;

end

Після обчислення контрольних входів, що використовують межі фізичного двигуна, контрольні входи вводяться в функцію динаміки квадрокоптера - quad\_dynamics\_nonlinear.m, для оновлення позиції та орієнтації квадрокоптера.

Останнє в циклі моделювання, функція plot\_quad.m визивається кожні три ітерації для побудови поточної позиції квадрата в тривимірному середовищі. Це дозволяє користувачеві візуалізувати поведінку квадрокоптера. Ця функція оновлює вершини чотирикутної рами та двигунів, використовуючи положення та орієнтацію квадрокоптера з рівнянь руху. Приклад цієї візуалізації показано нижче.

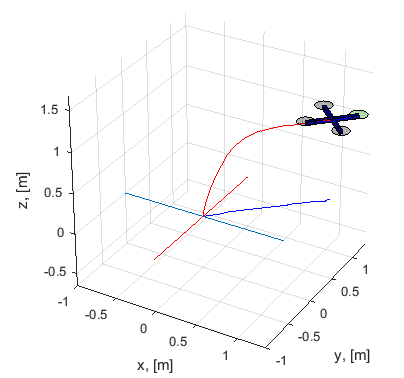


Рис. 36. Реакція квадрокоптера на одиничний вхід по чотирьом каналам керування

Після закінчення симуляції викликується функція plot\_data.m. Ця функція використовується для побудови різних показників ефективності квадрокоптера протягом часу моделювання. Сюди включаються фактичні та бажані положення та орієнтація ангелів, а також розрахункові вхідні дані. Ці ділянки можуть бути використані для ручного настроювання коефіцієнтів ПІД-контролера до певних характеристик продуктивності.

Графіки реакції системи на одиничний ступінчастий сигнал по трьом осям глобальної СК та по рисканню (Quad.X\_des = 1; Quad.Y\_des = 1; Quad.Z\_des = 1; Quad.psi\_des = pi/4) мають наступний вид:

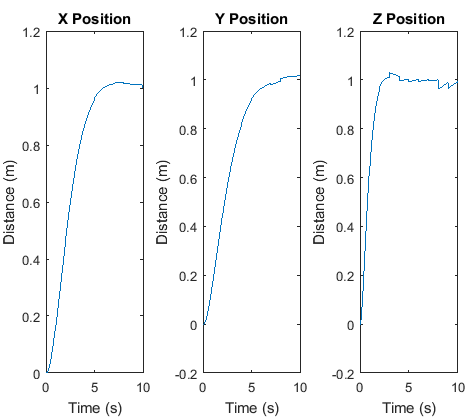


Рис. 37. Графіки перехідних процесів в глобальній СК

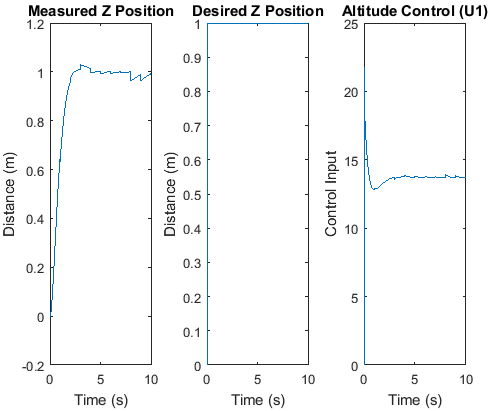


Рис. 38. Графіки перехідних процесів щодо каналу контролю висоти

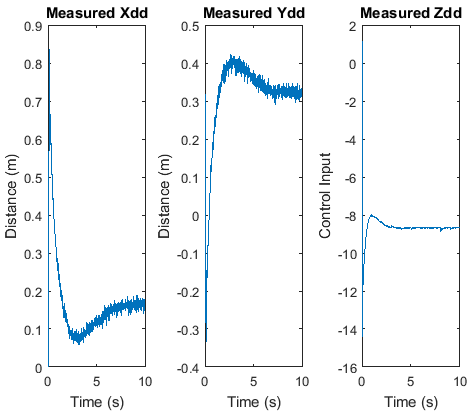


Рис. 39. Графіки вимірянних прискорень від сенсорів

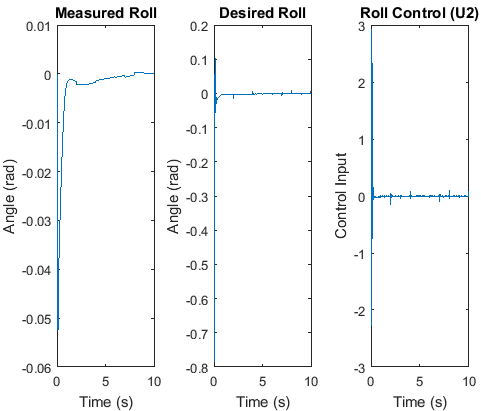


Рис. 40. Графіки перехідних процесів щодо контролю кута тангажа

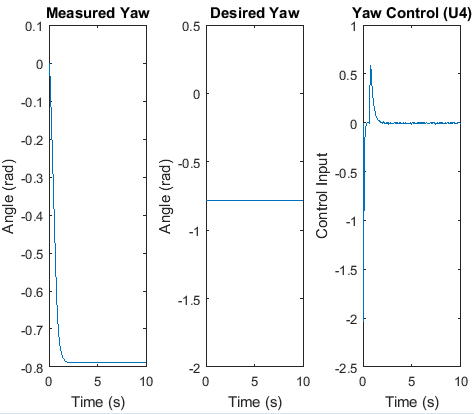


Рис. 41. Графіки перехідних процесів щодо контролю кута рискання (курс)

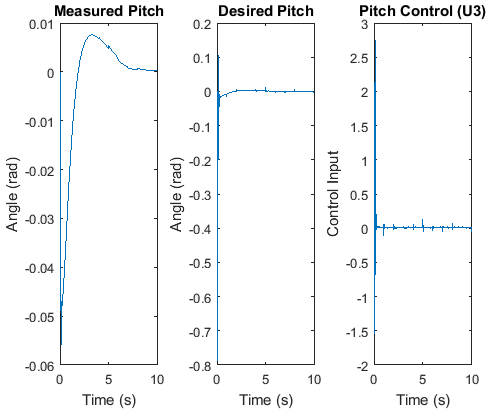


Рис. 42. Графіки перехідних процесів щодо контролю кута крену

**ВИСНОВКИ ДО П‘ЯТОГО РОЗДІЛУ**

Відомо, що квадрокоптер визначається набором нелінійних рівнянь, які роблять важким точне моделювання та контроль. Тому в даному розділі було спрощено рівняння руху квадрокоптера після прийняття декількох припущень щодо методу, який буде експлуатувано в квадрокоптері. Мета полягала в тому, щоб створити просту модель, але достатньо реальну.

Для підвищення схожості спрощеної моделі з реальною було впроваджено фізичні параметри квадрокоптера, основою якого був 3DR ArduCopter Quad-C Frame DIY. У даному розділі ми оцінили параметри моделі на основі нашої фізичної системи. Це допомогло визначити точні параметри для наших систем керування для швидковаого проведення початкової виставки.

Також, для поліпшення нашого моделювання і дизайну контролера, було розроблено модель імітації роботи датчиків, таких як гіроскоп та акселерометр. Дана розробка спиралась на відповідні дослідження з робіт [81, 82, 83], в яких проводились експериметни щодо знаходження похибок від даних датчиків. Результати моделювання представлено на рисунках 27 та 28.

Була проведена ідентифікація динаміки двигуна, знайдена її передатня функція шляхом проведення експериметну [78]. Зібравши необхідну інформацію про двигуни, було проаналізовано реальну роботу двигуна з нашою спрощеною моделею. Результат даного дослідження продемонстровано на рисунку 30.

Як завершаючий етап, була розроблена система керування БПЛА типу квадрокоптер засобами Matlab. Також, було проведено оцінку підвищення надісності ПІД-регуляторів після декількох модифікацій, які були представлені в роботі. Дані модифікації покращили надійність та стійкість системи, що дає нам можливість в подальшому тестувати дану систему керування на реальному фізичному тілі.

Результатом роботи даної системи є рисунки 36-41, на яких зображена реакція квадрокоптера на одиничний крок по чотирьом каналам керування. Точність стабілізації квадрокоптера в бажаному положені складає 4см.

1. **АВТОМАТИЧНА ПОСАДКА КВАДРОКОПТЕРА НА ОСНОВІ СИСТЕМИ ТЕХНІЧНОГО ЗОРУ**

В даному розділі буде здійнено опис алгоритму та реалізації засобами Matlab автоматичної посадки квадрокоптера на основі інформації від СТЗ.

Задача автоматичної посадки включає в себе 2 етапи:

1. 1 етап – наведення квадрокоптера на орієнтир;
2. 2 етап – зниження та посадка квадрокоптера на орієнтир.

**6.1. Алгоритм роботи наведення квадрокоптера**

При потраплянні орієнтира в об‘єктив оптичного датчика, СТЗ ідентифіковує його, знаходить положення орієнтира (*xo\_пікс, yo\_пікс*) на зображені (в піксельній СК зображення) та вираховує відстань (*, )* від центра (*xc\_ пікс, yc\_ пікс*) зображення до даного орієнтира.

*= xo\_пікс – xc\_пікс;* (6.1)

*= yo\_пікс – yc\_пікс;* (6.2)

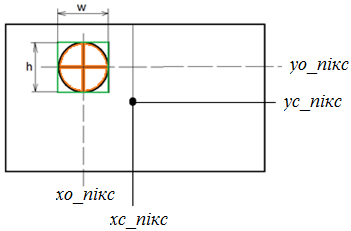


Рис. 43. Піксельна система координат на зображені

(6.3)

де – відстань до орієнтира від оптичної осі на зображені [пікселів].

Після цього розраховується земна відстань БПЛА від виявленого орієнтира в площині горизонту *l*:

, (6.4)

де GSD - розмір пікселя на поверхні землі [м / піксель],

В - розмір ширини зображення [піксель],

Н - висота БПЛА над поверхностю землі [м],

F - фокусна відстань [м],

b - розмір ширини матриці оптичного датчика [м],

K - оптичне збільшення оптичного датчика [1].

Також, маючи заздалегідь інформацію про розміри орієнтира (нехай маємо коло радіусом *R* = 1м) і необхідний розмір орієнтира на зображені (*Rпикс =* 10 пікселів), можно знайти відстань по вертикалі від орієнтира до БПЛА (висоту БПЛА). Алгоритм визначення висоти наступний:

З фомули (3.1) для камери, фізичний розмір матриці якої по діагоналі 1/3", a × b = 4.8 × 3.6 мм, а фокусна фідстань 3.6мм (оптичне збільшення К = 1):

(6.5)

(6.6)

(6.7)

(6.8)

При зміні розміру орієнтира на зображені, маємо інформацію про зміну відносну висоту БПЛА від цього ж орієнтира. Точність такого знаходження висоти при такій конфігурації камери досягає см.

Далі ці дані відправляються в автопілот для здійснення відповідних маневрів.

Основний принцип наведення квадрокоптера на орієнтир є мінімізація відстані між головною оптичною віссю камери та положення орієнтира на зображенні. Тобто, система керування приймає значення різниці між оптичною віссю камери і положенням знайденого орієнтира як значення помилки щодо бажаного положення квадрокоптера в глобальній СК.

СТЗ після кожного циклу обробки зображення видає нові значення дистанції та напрямку БПЛА до орієнтира, які коригують рух БПЛА до орієнтира.

* + 1. **Симуляція наведення квадрокоптера за даними системи технічного зору в програмному забезпечені Matlab.**

Було здійнено моделювання руху квадрокоптера в режимі наведення за даними СТЗ засобами Matlab. Суть моделювання та задумка наступні:

1. Нехай квадрокоптера має початкові задані координати стабілізації (0;0;20м).
2. Підключеною камерою до комп’ютера ми проводимо зйомку місцевості.
3. Представимо, що площина зйомки місцевості – відповідна площині по осях X та Y в середовищі Matlab, а ось Z – висота БПЛА, що відповідає відстані камери від орієнтира, що залежить від співвідношення з формули (6.8).
4. Тоді, коли СТЗ знайде орієнтир на місцевості, БПЛА почне здійснювати маневр в середовищі Matlab, яке відповідає положенню орієнтира щодо камера на реальній місцевості.

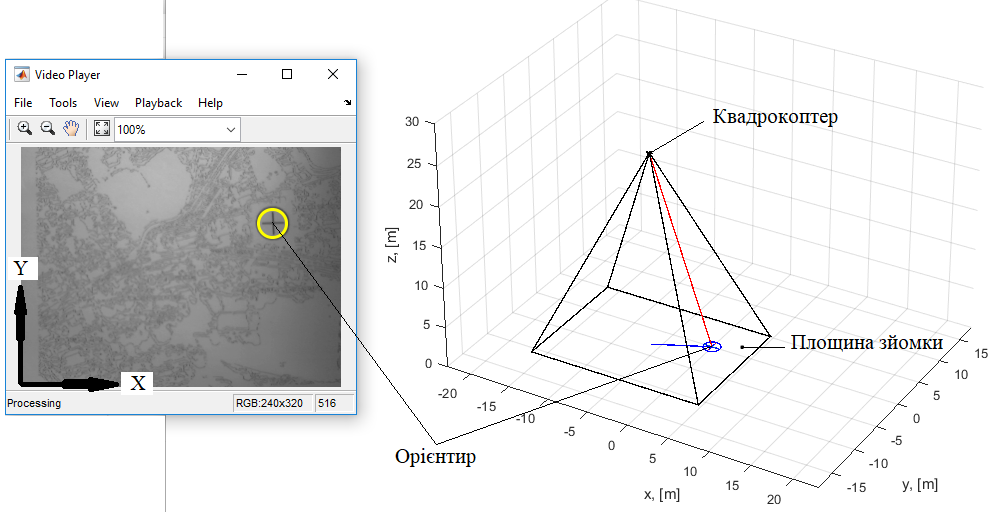


Рис. 44. Симуляція режиму наведення за даними СТЗ в програмному забезпечені Matlab

Симуляція проходить в режимі реального масштабу часу, тому представимо кінцеву траєкторію польоту БПЛА (червона лінія – політ по 3 осям; синя – проекція червоної лінії на площину XY – горизонт).

* 1. **Алгоритм роботи посадки квадрокоптера**

Алгоритм посадки квадрокоптера доповнює алгоритм наведення зміною висоти з наближенням до орієнтира посадки за певним законом зміни трєкторії. На рисунку 45 зображено варінти траєкторії посадки.

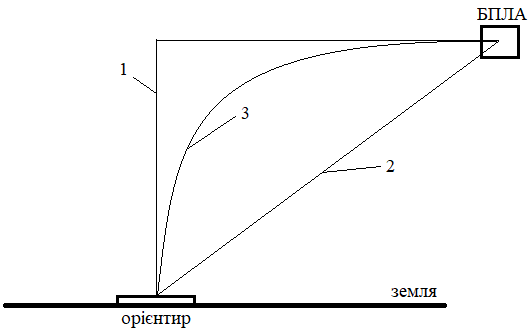


Рис. 45. Варіанти траєкторії посадки квадрокоптера

В задачі посадки БПЛА по даним СТЗ головним є максимально близько підлетіти до орієнтира посадки, а вже потім здійснювати зниження. Але, якщо почати знижуватися тільки тоді, коли положення орієнтира співпаде з оптичною віссю камери (траєкторія 1), то знижуєються швидкодія маневру в цілому. Якщо ж почати зниження разом із наближенням до орієнтиру (траєкторія 2), то встає питання щодо забезпечення надійності маневру, оскільки при такому русі орієнтир на зображенні залишається на тому ж місці, на якому був знайдений перший раз, а це, скоріше за все, на границі зображення. Тобто при певному зовнішньому збурені на БПЛА, орієнтир може бути втрачений. Тому за критерієм ефективності (оптимізація по швидкодії та надійності) накращим вибором траєкторії є рух по еліпсу (траєкторія 3).

* + 1. **Траєкторія посадки квадрокоптера**

Маючи приблизні значення положення орієнтира відносно квадрокоптера, ми можемо побудувати траєкторію посадки.

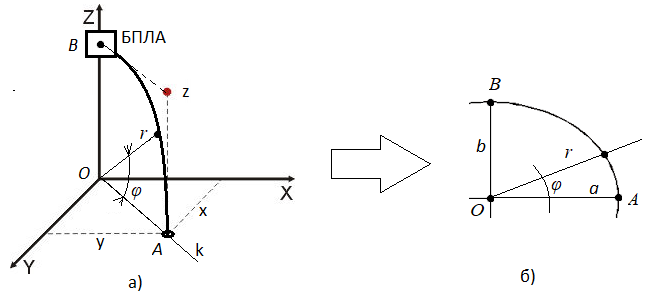


Рис. 46. Перехід від 3-ох вимірної моделі (а) побудови траєкторії до 2-х (б)

а) для побудови так званих точок шляху розглядаємо траєкторію в площині BOA;

б) коли осі ОА = = , а ОВ = *b =* , де , – початкові значення БПЛА щодо положення орієнтира. Використовуємо полярні координати:

*,* (6.2.1.1)

(6.2.1.2)

та формули еліпса

(6.2.1.3)

(6.2.1.4)

(6.2.1.4)

Таким чином маємо змінні (*x,y,z),* які відповідають параметрам положення орієнтира щодо БПЛА.

Дискретизація траєкторії відбувається при задання відповідних кутів точок траєкторії. Для посалки квадрокоптера з урахуванням захвату площини місцевості оптичного датчика достатнім є наступні значення:

*.*

Тоді, маємо 4 точки нашої траєкторії:

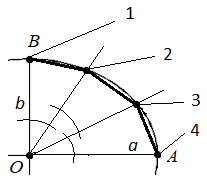


Рис. 47. Траєкторія шляху по еліпсу

На рисунку 47 цифрами зображені траєкторні точки.

* + - 1. **Посадка на малій висоті з урахуванням характеристик системи технічного зору**

З розділу 3.1 зрозуміло, що оптичний датчик захоплює меньшу ділянку землі при знижені висоти БПЛА, оскільки кут огляду залишається не змінним. Якщо маємо розмір орієнтира по одній стороні в 1 метр, то при оптичному збільшені K = 1(кут огляду 53°), мінімально можлива висота розпізнавання такого орієнтира досягає 1 метра. А це означає, що ми вже не можемо корегувати посадку квадрокопера лише за даними з СТЗ.

В даній роботі пропонується наступний метод вирішення даної проблеми. По даним СТЗ квадрокоптер підлітає до орієнтира по точкам траєкторії (з розділу 6.2.1), але остання точка має координати положення орієнтира щодо квадрокоптера на 2 метра вище. Це означає, що квадрокоптер по даним СТЗ зависне над орієнтиром на висоті в 2 метра. Реалізація алгоритму побудови модифікованої траєкторії з урахуванням оптичних можливостей СТЗ для посадки квадрокоптера засобами Matlab представлено в Додатку Е.

Результат даного алгоритму наступний:

>> [waypoints, psi\_des] = waypoints\_calc(15, 10, 20)

waypoints =

0 0 10.0000

7.4913 4.9942 12.4055

12.9854 8.6569 18.9896

15.0000 10.0000 28.0000

psi\_des = 0.5880

Psi\_des – кут курсу [рад], задає напрямок руху для квадрокоптера.

Контроль висоти відбувається як через інерціальну навігаційну систему (ІНС), так і через СТЗ. Досягнувши висоти 2 метри над орієнтиром, СТЗ починає працювати за іншим алгоритмом розпізнавання - «target in target», а ІНС корегує швидкість зниження та орієнтацію БПЛА.

Суть алгоритму розпізнавання «target in target» полягає у розпізнаванні орієнтиру таким чином, що залежно від висоти БПЛА виконується автоматична зміна алгоритму розпізнавання орієнтиру.

Наприклад, символ «Н», обведений колом. При великих дистанціях ми знаходимо круг, а при приближені камери до орієнтиру (при зменшені висоти *Н*), коло виходить за рамки захвату камери, і ми ідентифікуємо символ по його перетину ліній, як показано на рисунках 48 та 49.

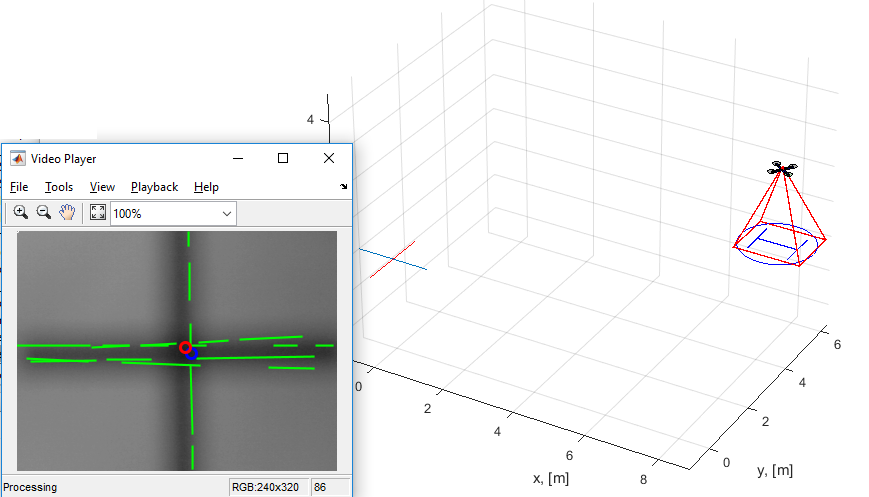


Рис. 48. Демонстрація виходу кола за межі зображення

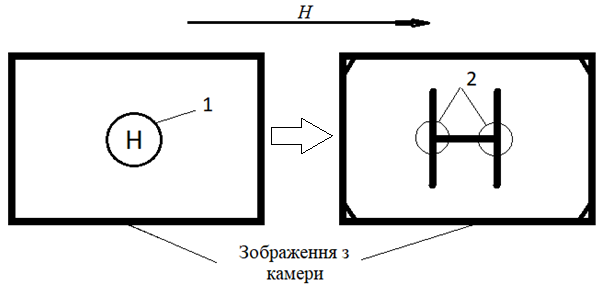


Рис. 49. Признаки ідентифікації символа при різній висоті *H* БПЛА: 1 – коло; 2 – перетин ліній.

* + 1. **Симуляція автоматичної посадки квадрокоптера за даними системи технічного зору в програмному забезпечені Matlab**

Було здійнено моделювання руху квадрокоптера в режимі посадка за даними СТЗ засобами Matlab. Суть моделювання та задумка наступні:

1. Стабілізація БПЛА з початковими координатами (0;0;20м).
2. Пошук орієнтиру оптичним датчиком.
3. Розпізнавання орієнтиру та визначення його координат.
4. Побудова траєкторії до орієнтиру.
5. Стабілізація БПЛА на висоті 2 м.
6. Виконання алгоритму «target in target».
7. Автоматична посадка БПЛА на орієнтир.

Симуляція проходить в режимі реального масштабу часу, тому представимо кінцеву траєкторію польоту БПЛА (червона лінія – політ по 3 осям; синя – проекція червоної лінії на площину *XY* – горизонт).

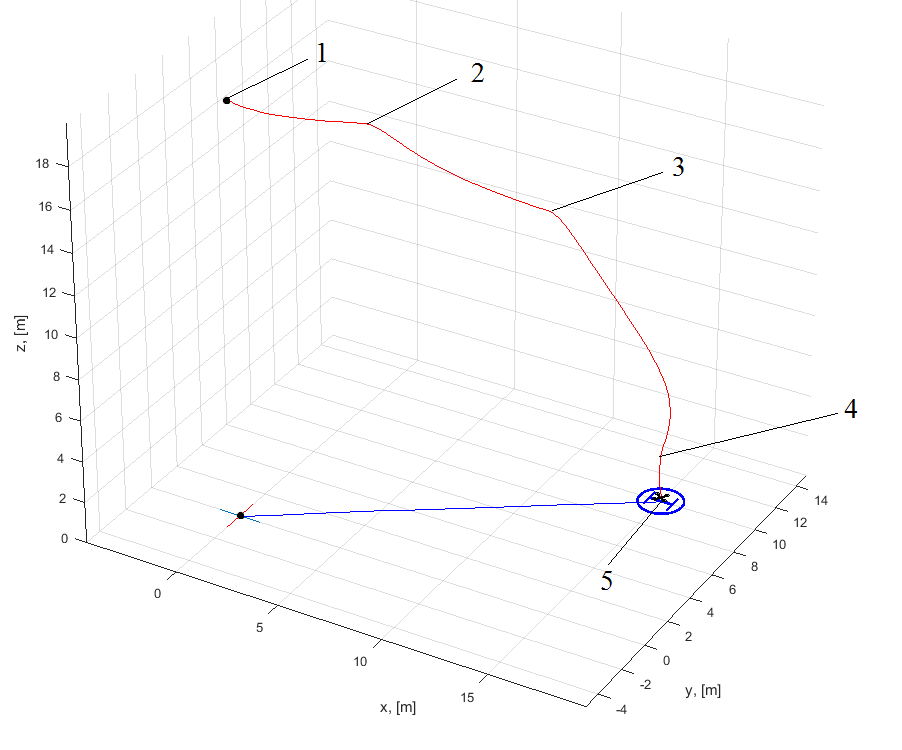


Рис.50.Симуляція автоматичної посадки квадрокоптера за даними СТЗ в програмному забезпечені Matlab

1 – початкові координати БПЛА;

2,3 – точки шляху траєкторії посадки;

4 – точка шляху траєкторії, в якій висота БПЛА відносно орієнтира досягла помітки в 2 метра;

5 – кінцева посадка БПЛА.

**ВИСНОВКИ ДО ШОСТОГО РОЗДІЛУ**

На основі розробленої системи керування БПЛА типу квадрокоптер з розділу 5 та розробленої ПО системи комп‘ютерного зору з розділу 4 в даному розділі була продемонстрована можливість роботи комбінації даних систем для вирішенння задач наведення та посадки в середовищі програмування Matlab.

За даним дослідженням оптимальною траєкторію посадки за критерієм ефективністі є форма еліпса, тобто її дуга.

Була вирішена проблема посадки на малій висоті, коли зовнішня рамка орієнтира виходить на межі діапазона огляду оптичного датчика, розробивши та використавши метод «target in target» разом з ІНС як додаткове джерело інформації.

За результатами проведеної симуляції розроблений алгоритм здійснює посадку БПЛА типу квадроктпер з похибкою в планарних координатах не більше 5см. На такий показник впливає не досліджені поки що похибки інерціально вимірювальних модулів ІНС.

1. **РОЗРОБЛЕННЯ СТАРАП-ПРОЕКТУ** 
   1. **Опис ідеї проекту**
      1. **Зміст ідеї**

Назва проекту:

Система автоматичної посадки «SkyLand» безпілотного літального апарату типу коптер на основі системи технічного зору.

Проект передбачає розробку системи автоматичної посадки «SkyLand» безпілотного літального апарату типу «коптер» на спеціально виділене місце без втручання людини. Дана система вирішує проблеми автоматизації посадки дрона для доставки певних об‘єктів та посадки під час аварійного режиму.

Таблиця 1. Опис ідеї стартап-проекту

|  |  |  |
| --- | --- | --- |
| Зміст ідеї | Напрямки застосування | Вигоди для користувача |
| Система автоматичної посадки «SkyLand» безпілотного літального апарату типу «коптер» | 1. Поштові сервіси 2. Транспортування товарів (медикаменти). 3. Навчання нових користувачів літальниї апаратів. | 1. Підвищена безпека повітряного простору. 2. Високоточний алгоритм автоматичної посадки з точністью до 0.1 метра. 3. Змога змінювати місце посадки (режим слідування) в реальному часі. 4. Персоналізація місця посадки свого літального апарату. |

* + 1. **Аналіз потенційних техніко-економічних переваг ідеї**

Таблиця 2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | Техніко-економічні характерис-тики ідеї | Товари/концепції конкурентів | | Мій проект | Слаб-ка сто-рона | Нейтраль-на сторона | Силь-на сто-рона |
| 3DR | AERO |
| 1. | Економічні: | Вартість витратних матеріалів, обслугову-вання: 130$. Наявність знижок. | Вартість витратних матеріалів, обслугову-вання: 135$.  Наявність знижок. | Вартість витратних матеріалів, обслугову-вання: 80$.  Знижки відсутні |  | + |  |
| 2. | Надійності:  Строк безвідмовної праці | 1 рік | 1 рік | 3 роки |  |  | + |
| 3. | Ергономічні: | Зручність використання при допоміжних операціяї | Вибір можливості сприйняття інформації користувачем | Відсутність відбрацій; раціональне розміщення;  простота набуття навичок |  | + |  |
| 4. | Естетичні | Розділена конструкція; відповід-ність стилю | Розділена конструкція; досконалість виробничого виконання | Раціональ-ність форми; цілісність композиції; стабільність товарного вигляду |  | + |  |
| 5. | Транспорта-бельність: трудомісткість підготовки | Середня | Середня | Незначна |  |  | + |
| 6. | Безпеки | Відсутність блокуючих пристроїв; експлуатація до 1 року | Відсутність блокуючих пристроїв; експлуатація до 1 року | Наявна блокуюча плата переванта-ження; експлуатація до 3 років |  |  | + |
| 7. | Технологічні | Налагодже-не виробництво з адаптивною технологією; оптимізація виготовле-ння та використа-ння товару; низька собівартість товару | Налагодже-не виробництво з адаптивною технологією; оптимізація виготовле-ння та використа-ння товару; низька собівартість товару | Висока собівартість товару; низьку потужності виробництва; жостка адаптація. | + |  |  |

* 1. **Технологічний аудит ідеї проекту**

Таблиця 3. Технологічна здійсненість ідеї проекту.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Ідея проекту | Технології її реалізації | Наявність технології | Доступність технологій |
| 1. | Ідентифікація орієнтира | Алгоритм системи компютерного зору. | Доробити алгоритм ідентифікації орієнтира на основі даної магістерської роботи | Доступні (власна розробка) |
| 2. | Інтеграція алгоритму ідентифікації цілі в мікрокомпютер | Загрузка та встановлення програмного забезпечення в мікрокомпютер | Відомі методи реалізації даної ідеї | Доступні (open sourse\*) |
| 3. | Конструкція системи технічного зору | Підбір відповідних елементів та закріплення на одній платформі | Доробити платформу даної системи | Доступні (власна розробка) |
| 4. | Елементи системи технічного зору | Підбір відповідних елементів з готових комерційних рішень | Доробити платформу даної системи | Доступні (власна розробка) |

Обрана технологія реалізації ідеї проекту: з готових комерційних рішень щодо мікрокомпютера та оптичного датчика зібрати готову до роботи систему технічного зору з власними алгоритмами роботи (ідентифікація цілі та побудови точок шляху посадки літального апарата).

* 1. **Аналіз ринкових можливостей запуску стартап-проекту**
     1. **Аналіз попиту**

Таблиця 4. Попередня характеристика потенційного ринку стартап-проекту.

|  |  |  |
| --- | --- | --- |
| № | Показник стану ринку (найменування) | Характеристика |
| 1 | Кількість головних гравців, од | 3 |
| 2 | Загальний обсяг продаж, грн/ум.од | 2904 |
| 3 | Динаміка ринку (якісна оцінка) | Зростає |
| 4 | Наявність обмеження для входу (вказати характер обмежень) | Реклама, лояльність покупців, ефект масштабу, хижацька цінова політика |
| 5 | Специфічні вимоги до стандартизації та сертифікації | Безпека для життя, охорона навколишнього середовища, якість послуги та обслуговування, гарантії виконавця послуги та ступінь довіри споживачів |
| 6 | Середня норма рентабельності в галузі (або по ринку), % | 12.9% |

Даний ринок є привабливим для входження за попереднім оцінюванням.

* + 1. **Потенційні групи клієнтів**

Таблиця 5. Характеристика потенційних клієнтів старап-проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Потреба, що формує ринок | Цільова аудиторія (цільові сегменти ринку) | Відмінності у поведінці різних потенційних цільових груп клієнтів | Вимоги споживачів до товару |
| 1 | Посадка в аварійному режимі | Користувачі-початківці. Вчителя курсів. | Простота інтеграції модуля в свої системи | Надійність |
| 2 | Точність наведення літального апарату на місце посадки | Юридичні компанії в сфері доставки товарів та послуг | Встановалення своїх орієнтирів для відповідних траєкторій | Надійність, точність, персоналізація місця посадки |

* + 1. **Аналіз ринкового середовища**

Таблиця 6. Фактори загроз

|  |  |  |  |
| --- | --- | --- | --- |
| № | Фактори | Зміст загрози | Можлива реакція компанії |
| 1 | Конкуренти | Штучне заниження цін на власне виробництво | Маркетингова компанія щодо піару свого проекту |
| 2 | Економічні чинники | Економічний спад в галузі | Переорієнтація на европейский ринок. |
| 3 | Демографія | Вузьке коло зацікавлених потенційних клієнтів в українському регіоні | Зацікавлення осіб віком до 45 років новим хоббі літальних апаратів. |
| 4 | Нормативно-правове середовище | Введення нових законів сертифікації та легалізації для продуктів літальних апаратів | Адаптація політики компанії щодо нововведень |

Таблиця 7. Фактори можливостей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Фактори | Зміст можливостей | Можлива реакція компанії |
| 1 | Іновації, в тому числі нові технології та Інтернет | Ноу-хау в сфері автоматизації посадки квадрокоптера | Збільшення об‘єму випуску товару з їх модернізацією |
| 2 | Партнери | Співпрацця з іншими компаніями даної галузі | Заключення договорів з компаніями-виробниками власних автопілотів для створення та випуску комплексних систем |
| 3 | Соціальні тенденції | Вподобання нових технологій для власного хоббі та проведення вільного часу | Поширення інформації про нові види зайнятості в даній галузі, провдення конкурсів авіо-любителів |

* + 1. **Аналіз пропозиції**

Таблиця 8. Ступеневий аналіз конкуренції на ринку

|  |  |  |
| --- | --- | --- |
| Особливості конкурентного середовища | В чому проявляється дана характеристика | Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною) |
| Олігополістична конкуренція | 2 великі компаній-конкурентів в даній сфері | Піар компанія щодо можливостей нашої продукції |
| Національна конкуренція | Конкуренція в масштабах країни | Охопити інші регіони регіони |
| Внутрішньогалузева конкуренція | Товар для вузького кола користувачів | Реалізація нових ідей |
| Товарно-видова конкуренція | Наявність вже готових аналогів відомих брендів | Демонстраційні заходи щодо показу своїх товарів |
| Цінова конкуренція | Штучне заниження ціни | Показ єфективності та надійності нашої системи |
| Марочна конкуренція | Наявність міжнародніх відомих брендів | Створення власного бренду |

* + 1. **Детальний аналіз умов конкуренції в галузі**

Таблиця 9. Аналіз конкуренції в галузі за М. Портером

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Складові аналізу | Прямі конкуренти в галузі | Потенційні конкуренти | Постачальники | Клієнти | Товари-замінники |
| 3DR  AERO | Вхід в галузь компанії Raspberry Pi Foundation | JeVois – компанія середнього рівня, розрахована на 1632 одиниць продукції в місяць.  Asus – компанія міжнародного рівня. | Радіо-любителі.  Поштові служби Amazon.  Компанії медичної орієнтації | Система OptFlow від 3DR – модернізація даного продукту. |
| Висновки | Значних складнощів не повинно бути: їх спеціалізація в основному лише є автопілоти. | Є всі можливості входу в ринок. По заяві від компанії планується через 10 місяців. | Маємо сильних та надійних постачальників, які мають суттєвий вплив на ринок елементів. | В цілому - так. Надійність та ціна товару. | Певні складнощі |

З оглаяду на конкурентну ситуацію існує принципова можливість для виходу на ринок в даній галузі. Проект повинен бути надійним та с легкою інтеграцією в інші автопілоту з логічно зрозумілом інтерфейсом настроювання.

* + 1. **Визначення та обгрунтування факторів конкурентоспроможності**

Таблиця 10. Обгрунтування факторів конкурентоспроможності

|  |  |  |
| --- | --- | --- |
| № | Фактори конкурентоспроможності | Обгрунтування |
| 1 | Надійність системи | Здатність системи в аварійному режимі здійснити посадку в будь-якій рівній безпечній місцевості |
| 2 | Простота інтеграції в інші системі | Підкючення лиша декількому кабелями, скачати прошивку для автопілота, закріпити на корпусі літального апарата |
| 3 | Персоналізація орієнтира користувачем | Інтелектуально логічний інтерфейс настройки своїх орієнтирів для різних задач користувача |
| 4 | Ноу-хау в сфері достаки товарів | Нові методи використання дронів |

* + 1. **Аналіз сильних та слабких сторін старап проекту**

Таблиця 11. Порівняльний аналіз сильних та слабких сторін «SkyLand».

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Фактори конкурентоспроможності | Бали  (1 – 20) | Рейтинг товарів конкурентів у порівнянні з «SkyLand» | | | | | | |
| -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| 1 | Надійність системи | 16 |  |  |  |  |  | + |  |
| 2 | Простота інтеграції | 16 |  |  |  |  |  | + |  |
| 3 | Персоналізація орієнтира | 19 |  |  |  |  |  |  | + |
| 4 | Ноу-хау в сфері достаки товарів | 15 |  |  |  |  | + |  |  |
| 5 | Адаптація під нові запроси ринка | 5 |  | + |  |  |  |  |  |

* + 1. **Складання SWOT-аналізу**

Таблиця 12. SWOT-аналіз стартап-проекту

|  |  |
| --- | --- |
| Сильні сторони: персоналізація системи; нижча ціна товару; простота інтеграції в автопілоти. | Слабкі сторони: низька спроможність адаптації щодо потреб ринку; незначні потужності виробництва; висока собівартість. |
| Можливості: почати виготовлення складових товару самостійно для зниження собівартості виробництва; піар компанія нового товару з новими можливостями для звичайного користувача; вигідні домовленості про зниження ціни на поставку сировини. | Загрози: штучне заниження ціни компаніями-конкурентів; низька популярність товару; недовіра клієнта; низька фінансова підтримка проекту; нові розробки компаній-конкурентів. |

* + 1. **Альтернативи ринкової поведінки**

Таблиця 13. Альтернативи ринкового впровадження старап-проекту

|  |  |  |  |
| --- | --- | --- | --- |
| № | Альтернатива (орієнтований комплекс заходів) ринкової поведінки | Ймовірність отримання ресурсів | Строки реалізації |
| 1 | Стратегія мінімальних витрат: підвищення ефективності керування внутрішнім  ланцюжком вартості шляхом інноваційного вдосконалення бізнесмоделі підприємства і використання резервів альтернатив зниження витрат в окремих її ланках;  накопиченому досвіді, економії від організації придбання ключових ресурсів, враховуючи можливості найму дешевої робочої сили, економію на крупних за обсягом закупівлях партій матеріалів, сировини, регіональних відмінностях за окремими статтями витрат тощо;  виборі та дотриманні політики піонера або наслідувальника. Реорганізації ланцюжка вартості підприємства в різних формах альтернативах. | 50% | 23 місяці |
| 2 | Стратегія диференціації за різними параметрами: різноманітність споживчих властивостей, більший вибір та просту процедуру придбання, унікальний сервіс, термінове постачання,  підвищену споживчу цінність при старій ціні, нові товари, унікальний дизайн, престижність та неповторність, надійність і безпечність, якість виконання, сервіс, імідж і репутацію, широкий асортимент.  Збільшення різновидів моделей товару, скорочення часу розробки нових моделей товару, забезпечення лідерства в пропозиції нових товарів на основі можливості переробки. | 5% | 38 місяців |
| 3 | Стратегія оптимізації витрат: досягнення оптимально низьких витрат у поєднанні із обмеженою диференціацією в бізнесі, зорієнтованому своїм продуктом на чутливих до ціни споживачів. Інвестування в нові технології, створення автономної системи дилерів продукції, індивідуальне обслуговування, запровадження електронної торгівлі. | 15% | 17 місяців |
| 4 | Стратегія утвередження (створення іноваціійної цінності): стратегія щодо обрання моделі інноваційної поведінки підприємств > стратегія імітування > вид "залежності" та "вдосконалення". Субстратегія "технологічні інновації (розробки, проекти)". Альтернативи: вдосконалення діючої технології; підтримка діючої технології; використання чужої технології.  Забезпечення зниження собівартості продукції за рахунок досконаліших матеріальнотехнічних ресурсів та їх ефективного використання, а також більш якісних виробничо технологічних процесів;  забезпечення нарощування обсягів виробництва та продажу продукції, яка користується попитом, на діючих або на новостворюваних і продуктивніших потужностях;  створення нових умов для виробництва нових, конкурентоспроможних продуктів. | 30% | 9 місяців |

З таблиці 13 робимо висновок, що стратегія 1 є більш простим і ймовірним щодо отримання ресурсів, а стратегія 4 має найбільш стислі строки реалізації.

* 1. **Розроблення ринкової стратегії проекту**
     1. **Визначення стратегій охоплення ринку**

Таблиця 14. Вибір цільових груп потенційних споживачів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Опис профілю цільової групи потенційних клієнтів | Готовність споживачів сприйняти продукт | Орієнтовний попит в межах цільової групи (сегменту) | Інтенсивність конкуренції в сегменті | Простота входу у сегмент |
| 1 | Звичайні користувачі | Так | Радіолюбителі віком від 16 до 60 років. | Низька | Легко |
| 2 | Поштобі служби України | Так | Укрпошта, Нова пошта | Не існує | Складно |
| 3 | Медичні заклади України | Так | Державні установи медичної допомоги | Не існує | Складно |

З огляду на цільові групи потенційних клієнтів приймаємо рішення використовувати масовий маркетинг, оскільки конкуренція в напрямках медичних закладів і почтових служб практично відсутні на території України. Щодо звичайного користувача є деякі складнощі в просувані продукту, оскільки мало українців мають інтерес до авіомоделей.

* + 1. **Базова стратегія розвитку**

Таблиця 15. Визначення базової стратегії розвитку

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Обрана альтернатива розвитку | Стратегія охоплення ринку | Ключові конкурентоспроможні позиції відповідно до обраної альтернативи | Базова стратегія розвитку |
| 1 | Вдосконалення діючої технології | Цільовий, чи концентрований маркетинг | Задовольняти потреби обраного сегмента ринку | Стратегія односегментної концентрації: - спеціалізація діяльності фірми на потребах певного ринкового сегмента;  - спеціалізація на одному сегменті забезпечує економію завдяки концентрації коштів. |
| 2 | Підтримка діючої технології | Цільовий, чи концентрований маркетинг | Задовольняти потреби обраного сегмента ринку | Стратегія сегментної спеціалізації: - зосередження на потребах певної групи споживачів, можливість їх широкого задоволення й досягнення позитивного іміджу серед даного сегмента ринку;  - можливість підвищити увагу до інших різновидів товару. |

* + 1. **Стратегія конкурентної поведінки**

Таблиця 16. Визначення базової стратегії конкурентної поведінки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Чи є проект «першопрохідцем» на ринку | Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів? | Чи буде компанія копіювати основні характеристики товару конкурента, і які? | Стратегія конкурентної поведінки |
| 1 | В Україні – так | Нові споживачі | Ні | Стратегія лідера |
| 2 | В світі – ні | Забирати у конкурентів | Ні | Стратегія заняття конкурентної ніші |

* + 1. **Стратегія позиціонування**

Таблиця 17. Визначення стратегії позиционування

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вимоги до товару цільової аудиторії | Базова стратегія розвитку | Ключові конкуренто-спроможні позиції власного стартап-проекту | Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових) |
| 1 | Персоналізація орієнтирів посадки;  Простота інтеграції | Стратегія односегментної концентрації | Задовольняти потреби обраного сегмента ринку | Надійність, практичність, персоналізаційність. |
| 2 | Надійність системи: безпечна апарійна посадка | Стратегія сегментної спеціалізації | Задовольняти потреби обраного сегмента ринку | Надійність, практичність, персоналізаційність. |

* 1. **Розроблення маркетингової програми стартап-проекту**
     1. **Маркетингова концепція товару**

Таблиця 18. Визначення ключових переваг концепції потенційного товару

|  |  |  |  |
| --- | --- | --- | --- |
| № | Потреба | Вигода, яку пропонує товар | Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити) |
| 1 | Надійність | Автоматична посадка на безпечну місцевість | Можливість посадки на свою запрограмовану площадку |
| 2 | Персоналізація | Вибір користувачем свого власного орієнтира | Обір за бажанням своїх власних орієнтирів для контролю літального апарату |
| 3 | Інтегрованість | Проста | Автоматична прошивка ПО, підключення 3 кабелями з універсальними роз‘ємами |
| 4 | Політика цін | Гнучка | Дешевше товарів конкурента |

* + 1. **Трирівнева маркетингова модель товару**

Таблиця 19. Опис трьох рівнів моделі товару

|  |  |  |  |
| --- | --- | --- | --- |
| Рівні товару | Сутність та складові | | |
| Товар за задумом | Надійність: можливість посадки на свою запрограмовану площадку.  Персоналізація: обір за бажанням своїх власних орієнтирів для контролю літального апарату.  Інтегрованість: автоматична прошивка ПО, підключення 3 кабелями з універсальними роз‘ємами.  Ціна: дешевше товарів конкурента. | | |
| Товар у реальному виконанні | Властивості/характеристики | М/Нм | Вр/Тх/Тл/Е/Ор |
| 1. Колір (черний або сірий з повним покриттям). 2. Зовнішній вигляд (естетичний прямокутний вигляд). 3. Термін придатності: 5 років. 4. Рівень вібрації: 6 g. | Нм  М  М  М | Тл/Ор  Тл/Ор  Тл/Ор  Тл/Ор |
| Якість: тестування – в 99.986% літальний апарат здійснював посадку без позкоджень. Відповідає нормам ДСТУ ГОСТ 1.3:2005. | | |
| Пакування: упаковка флоу-пак з нанесенням спеціального дизайну. На упаковці міститься така інформація (маркування):   * загальна назва продукту, власна назва; * номінальну масу нетто, г; * кінцевий термін реалізації або дату виготовляння та термін придатності; * умови зберігання; * позначення ДСТУ 4503:2005; * найменування та адреса виробника і місце виготовлення; * товарний знак; * штриховий код згідно з ДСТУ 3147. | | |
| Марка: ТМ SkyFly – зареєстрована ТМ. «SkyLand» - торгова марка. | | |
| Товар із підкріпленням | В торгових мережах для споживачів періодично для стимулювання попиту діють спеціальні пропозиції, які дозволяють придбати товар зі знижкою або кілька одиниць товару разом за меншою вартістю тощо. Також інформацію про корисні властивості продукту споживач може отримати з інформаційних проспектів торгових мереж, зовнішньої реклами, реклами на телебаченні тощо. | | |

Проект буде захищено за рахунок захисту ідеї товару (захист інтелектуальності власності – патент) разом із іноваційним ноу-хау.

* + 1. **Визначення цінових меж**

Таблиця 20. Визначення меж встановлення ціни

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Рівень цін на товари-замінники | Рівень цін на товари-аналоги | Рівень доходів цільової групи споживачів | Верхня та нижня межі встановлення ціни на товар/послугу |
| 1 | Звичайні користувачі: $140 | $160 | $500+ | $75-135 |
| 2 | Поштобі служби України і медичні заклади України: $350 | $280 | $2000+ | $325-540 |

* + 1. **Оптимальна система збуту**

Таблиця 21. Формування системи збуту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Специфіка закупівельної поведінки цільових клієнтів | Функції збуту, які має виконувати постачальник товару | Глибина канала збуту | Оптимальна система збуту |
| 1 | Звичайні користувачі: Обсяг збуту – невеликі: поштучна закупівля.  Контакти з виробником – дуже тісні.  Політика цін – гнучка.  Право власності на товар – у посередника. | Транспортування, встановлення контакту, інформування, стимулювання руху товарів до споживача, проведення рекламних заходів, розпродажу | Канал першого рівня | Інтенсивний розподіл |
| 2 | Поштобі служби України і медичні заклади України: масова закупівля.  Знання предмета побуту – відмінні.  Зона дії – вузька.  Право власності на товар – у виробника. | Прийняття на себе ризику торгових угод. Участь у формуванні ціни на товари, що продаються. Маркетингові дослідження (наприклад, попиту споживачів), а також збирання відповідної інформації.  Встановлення безпосередніх контактів із споживачами та покупцями товарів.  Стимулювання руху товарів до споживача, проведення рекламних заходів, розпродажу. | Канал другого рівня | Екслюзивний розподіл |

* + 1. **Розроблення концепції маркетингових комунікацій**

Таблиця 22. Концепція маркетингових комунікацій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Специфіка поведінки цільових клієнтів | Канали комунікацій, якими користуються цільові клієнти | Ключові позиції, обрані для позиціонування | Завдання рекламного повідомлення | Концепція рекламного звернення |
| 1 | Звичайні користувачі: економні, байдужі. | Рекламне охоплення, частота, кількість контактів, телебачення, інтернет, журнали, газети, зовнішня реклама. | Рекламні радіо-кампанії; зображення-ілюстрації; | Притягнути увагу слухача з перших секунд, не даючи шансів перемкнутися на іншу програму.  Повідомлення мають чіткі факти, що запам'ятовуються, бути досить яскравим, щоб включати уяву слухача. | Привернути півищену увагу; легко читаним, містив більше фактів і поменше голослівних тверджень,  заголовок повинен привертати увагу, інформувати про товар і давати аргументи для купівлі; |
| 2 | Поштобі служби України і медичні заклади України: раціональні, благополучні та успішні | Рекламне охоплення,  інтернет,  контактів, телебачення. | Прохідність в місці демонстрації; велика кількість розміщень | Рекламу повинні побачити як можна більша кількість людина | Використання свідчення реальних людей, а краще — відомих експертів, що викликають довіру клієнтів. |

**ВИСНОВКИ ДО СЬОМОГО РОЗДІЛУ**

В даному розділі було розглянуто питання щодо доцільності і можливості створення старап проекту на основі даної магістерської роботи. Результатом даного дослідження існує достатньо велика можливість успішної реалізації проекту «SkyLand», в якому середня рентабельність може скласти приблизно 13%.

Існує перспектива впровадження з огляду на потенційні групи клієнтів, бар‘єри входження, стан конкуренції, конкурентоспроможність проекту.

Основна технологія виготовлення системи складається з готових комерційних рішень щодо мікрокомпютера та оптичного датчика, які в свою чергу зібрати в готову до роботи систему технічного зору з власними алгоритмами роботи (ідентифікація цілі та побудови точок шляху посадки літального апарата).

Було обрано стратегію утвередження (створення іноваціійної цінності): стратегія щодо обрання моделі інноваційної поведінки підприємств через стратегію імітування, яка в свою чергу через вид "залежності" та "вдосконалення". Субстратегія – "технологічні інновації (розробки, проекти)", вдосконалення діючої технології, підтримка діючої технології, використання чужої технології.

З оглаяду на конкурентну ситуацію існує принципова можливість для виходу на ринок в даній галузі. Проект повинен бути надійним та с легкою інтеграцією в інші автопілоту з логічно зрозумілом інтерфейсом настроювання.

**ВИСНОВКИ ПО РОБОТІ**

В магістерській дисертації було розроблено систему автоматичного керування безпілотного літального апарату за допомогою системи технічного зору для вирішення задачі його автоматичної посадки на спеціально підготовлену площадку в реальному масштабі часу з заданою точністю.

Були досягнуті наступні результати:

1) розроблено алгоритми автоматичного керування БПЛА типу квадрокоптер на основі інформації з СТЗ та методів інтеграції СТЗ в систему керування БПЛА;

2) розроблено комбінований метод ідентифікації місця посадки, оснований на декількох методів розпізнавання, які мають різні принципи роботи; в порівнянні з аналогічними проектами, дана система дає можливість ідентифікувати орієнтир навіть при його втраті;

3) підібрано та введено різні фільтри підвищення якості зображення, в тому числі фільтр Калмана для підвищення надійності розпізнавання орієнтиру, а також, щоб вилучити можливість раптового зникнення орієнтира із зображення;

4) розроблено новий багаторівневий алгоритм розпізнавання орієнтира «target in target» для вирішення проблеми посадки БПЛА на малих висотах. Суть алгоритму розпізнавання «target in target» полягає у розпізнаванні орієнтиру таким чином, що залежно від висоти БПЛА виконується автоматична зміна алгоритму розпізнавання орієнтиру. Працездатність даного методу підтверджено в даній роботі. За результатами проведеної симуляції розроблений алгоритм здійснює посадку БПЛА типу квадрокоптер з похибкою в планарних координатах не більше 5см. На такий показник впливає не досліджені поки що похибки інерціально вимірювальних модулів ІНС.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Техническое зрение в системах управления мобильными объектами-2010: Труды научно-технической конференции-семинара. Вып. 4 / Под ред. Р. Р. Назирова.— М. : КДУ, 2011.— 328 с. : табл., ил., цв. ил.
2. *Хорн Б*. К.П. Зрение роботов. – М.: Мир. – 1989. – 400 c.
3. *Колобородов В.Г.* Застосування методів і алгоритмів цифрової обробки зображень в оптико-електронних приладах / *В.Г. Колобородов, К.В. Харитоненко* // Вісник НТУУ “КПІ”. – К. : НТУУ “КПІ”, 2010. – № 40. – С. 23–31.
4. *Слюсар В.И.* Передача данных с борта БПЛА: стандарты НАТО / *В.И. Слюсар* // Электроника: наука, технология, бизнес. – 2010. – № 3. – С. 80–86.
5. *Слюсар В.И.* Радиолинии связи с БПЛА: примеры реализации / *В.И. Слюсар* // Электроника: наука, технология, бизнес. – 2010. – № 5. – C. 56–60.
6. *Кузнецов В.* Беспилотная одиссея в небе будущего / *В.Кузнецов* // Наука и техника. – Харьков, 2011. – № 5 (60). – С. 21–26.
7. Sylvester, David Tom. *Development of a Vision Based Landing System for Use on a Rotary Wing Unmanned Aerial Vehicle.* Sheffield, UK : Department of Automatic Control & Systems Engineering, The University of Sheffield, 2012.
8. Mbaekube, Ugonna. *Altitude Control and Stabilization of a Quadrotor.* Sheffield, UK : Department of Automatic Control & Systems Engineering, The University of Sheffield, 2012.
9. *Vision Guided Landing of an Unmanned Air Vehicle.* Omid Shakernia, Yi Ma, T. John Koo, Joao Hespanha, S. Shankar Sastry. Phoenix, Arizona : Proceedings of the 38th Conference on Design & Control, 1999.
10. Hintze, Joshua. *Autonomous Landing of a Rotary Unmanned Aerial Vehicle in a Non-Cooperative Environment Using Machine Vision.* Department of Electrical and Computer Engineering, Brigham Young University. 2004. Thesis.
11. *Vision Based Terrain Recovery for Landing Unmanned Aerial Vehicles.* Marci Meingast, Christopher Geyer, Shankar Sastry. Atlantis, Paradise Island, Bahamas : 43rd IEEE Conference on Decision and Control, 2004.
12. *Vision-based Autonomous Landing of an Autonomous Aerial vehicle.* S. Saripalli, J.F. Montgomery and Gaurav S. Sukhatme. Washington D.C. : in Proceedingsnce on Robotics & Automation of the 2002 IEEE International Confere, May 2002.
13. Vilas K. Chitrakaran, Darren M. Dawson, Jian Chen, and Matthew Feemster Member, IEEE. Vision Assisted Autonomous Landing of an Unmanned Aerial Vehicle. *IEEE explore.* [Online] 12-15 December 2005. [Cited: 01 August 2012.]

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1582365.

1. S. Saripalli, J.F. Montgomery, G.S. Sukhatme. Visually Guided Landing of an Unmanned Aerial vehicle. *IEEE transactions on Robotics and Automation.* June 2003, Vol. 19, Vol. 3.
2. E.D. Dickmanns, F.R. Schell. *Autonomous Landing of Airplanes by Dynamic Machine Vision.* Munchen, Germany : Universitat der Bundeswehr.
3. Omead Amidi, Takeo Kanade, Keisuke Fujita. A Vision Odometer for Autonomous Helicopter Flight. *Robotics and Autonomous Systems 28.* 1999.
4. Amidi, Omead. *An Autonomous Vision-Guided Helicopter.* Pittsburg : PhD Thesis Submitted to the Department of Electrical and Computer Engineering, Carnegie Mellon University, 1996.
5. T.J. Koo, F. Hoffmann, H. Shim, B. Sinopoli and S. Sastry. *Hybrid Control of an Autonomous Helicopter.* Berkeley, USA : Department of Electrical Engineering and Computer Sciences, University of California.
6. Omid Shakernia, Yi Ma, T. John Koo, Shankar Sastry. *Landing an Unmanned Air Vehicle: Vision Based Motion Estimation and Nonlinear Control.* s.l. : Asian Journal of Control, 1999. Vol. 1, No. 3, pp. 128 - 145.
7. C. Sharp, O. Shakernia, and S. Sastry. *A Vision System for Landing an Unmanned Aerial Vehicle.* Seoul, South Korea : Proceedings of the 2001 IEEE, 2001.
8. *A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves.* F. Mokhtarian, and F. Mohanna. 8, s.l. : IEEE transactions on pattern analysis and machine intelligence, 1992, Vol. 14.
9. *Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of Support.* Yung, X.C. He and N.H.C. Hong Kong : Department of Electrical and Electronic Engineering.
10. *Corner detector based on global and local curvature properties.* Yung, Xiao Chen He and Nelson H.C. 5, Hong Kong : Optical Engineering, May 2008, Vol. 47.
11. Rosenfield, L. Kitchen and A. Gray Level corner detection. *Pattern Recognition Letters, pp. 95 - 102.* 1982.
12. *Determination of ego-motion from matched points.* Harris, C. Cambridge, UK : : In Proceedings 3rd Alvey Vision Conference, 1987.
13. *Determination of ego-motion from matched points.* Harris, C. Cambridge, UK : In Proceedings 3rd Alvey Vision Conference, 1987.
14. *A new approach to low-level image processing.* Brady, S. Smith and J. 1, s.l. : International Journal of Computer Vision, 1997, Vol. 23.
15. *Vision-Based Autonomous Landing of an Unmanned Aerial Vehicle.* S. Saripalli, J.F. Motgomery, G. Sukhatme. Washington D.C. : Proceedings of the 2002 IEEE International Conference on Robotics and Automation, May 2002.
16. *A Learning Algorithm for Boltzmann Machines.* David H. Ackley, Geoffrey E. Hinton and Terrence J. Sejnowski. 9, s.l. : Cognitive Science 9, 1985.
17. Choi, Jay Hyuk. *Vision-based Automatic Landing on a Moving Target for Unmanned Helicopters*. Daejeon, South Korea: Korea Advanced Institute of Science and Technology. 2012.
18. Olalekan Ogunmolu. *AUTONOMOUS NAVIGATION OF A ROTOR-CRAFT UNMANNED AERIAL VEHICLE USING MACHINE VISION*. England, Sheffield: The University of Sheffield. August 2012.
19. Кузнецов В. Беспилотная одиссея в небе будущего / В.Кузнецов // Наука и техника. – Харьков, 2011. – № 5 (60). – С. 21–26.
20. Проценко М.М. Аналіз структури та варіантів побудови безпілотних авіаційних комплексів / М.М. Проценко // Вісник ЖДТУ. – Житомир : ЖДТУ, 2012. – № 61(2). – С. 113–117.
21. Харченко В.П., Чепіженко В.І., Тунік А.А., Павлова С.В. – А 205 Авіоніка безпілотних літальних апаратів; За ред. В.П. Харченка. – К.: ТОВ «Абрис-принт», 2012
22. С.Н. Безрядин. Основной недостаток сенсоров современных цифровых камер. 05.11.2003 г., www.zenit.istra.ru/articles/colour (05.2005)
23. В.В. Ростопчин. ПРИМЕНЕНИЕ ЦИФРОВЫХ ОПТИЧЕСКИХ СИСТЕМ ДЛЯ БЕСПИЛОТНЫХ ЛЕТАТЕЛЬНЫХ АППАРАТОВ / В.В. Ростопчин, М.Л. Дмитриев // ЦНИИ АРКС – Харьков, 2010. - – № 53. – С. 7
24. Цепляєва Т.П. Метод выбора характеристик фотооборудования для БПЛА в зависимости от высоты полета / Т.П. Цепляєва, А.Н. Лохов // Открытые информационные и компьютерные интегрированные технологии – Харьков, 2011. – № 49. – С. 48–52.
25. Розпізнавання образів та аналізування зображень [Електронний ресурс] / Студопедия.Орг - 2014-2016 // Дата публікувания: 11.12.2014 — Режим доступа: <http://studopedia.org/6-105312.html>
26. Техническое зрение в системах управления мобильными объектами-2010: Труды научно-технической конференции-семинара. Вып. 4 / Под ред. Р. Р. Назирова.— М. : КДУ, 2011.— 328 с. : табл., ил., цв. ил.
27. Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2005. – 1072 с.
28. Технічне довідкове керівництво [Електронний ресурс] – Firefly MV // Point Grey inc. - // Canada – 2011 – ст. 13. <https://www.ptgrey.com/support/downloads/10116>
29. COMPUTER VISION AND DEEP LEARNING [Електронний ресурс] - ВСТРАИВАЕМЫЕ СИСТЕМЫ -// МОДУЛЬ JETSON TK1. – NVidia inc. - <http://www.nvidia.com.ua/object/jetson-tk1-embedded-dev-kit-ru.html>
30. Вікіпедія – вільна енциклопедія [Електронний ресурс] - <https://uk.wikipedia.org/wiki/Lab>
31. Кузовков Н.Т., Салычев О.С. Инерциальная навигация и оптимальная фильтрация. – М.: Машиностроение, 1982. – 216с.
32. Grewal M.S., Andrews A.P. Kalman filtering: theory and practice using MATLAB. 3rd ed. J. Wiley & Sons, Inc. – 2008.
33. Jekeli C. Inertial Navigation Systems with Geodetic Application. Walter de Gruyter, Berlin, 2001.
34. Salychev O.S. Applied Inertial Navigation: Problems and Solutions. BMSTU Press, Moscow, Russia, 2004.
35. Titterton D.H., Weston J.L. Strapdown Inertial Navigation Technology, - American Institute of Aeronautics and Astronautics & Institution of Electrical Enginners, 2004.
36. Grewal M.S., Weill L.R., Andrews A.P. Global Positioning Systems, Inertial navigation and integration, 2nd ed., Wiley, New York, 2007.
37. Huddle J. Inertial Navigation System Error Model Considerations in Kalman Filtering Applications. In Control and Dynamic Systems. Vol. 20, Academic Press, 1983.
38. Faraz M. Mirzaei, Stergios I. Roumeliotis. A Kalman Filter – Based Algorithm for IMU – Camera Calibration: Observability Analysis and Performance Evaluation.
39. Баклицкий В.К. Методы фильтрации в корреляционно – экстремальных системах навигации. – М.: Радио и связь, 1986.– 216с.
40. Веремеенко К.К., Красильщиков М.Н., Сыпало К.А. и др. Управление и наведение беспилотных маневренных летательных аппаратов на основе современных информационных технологий. – М.: ФИЗМАТЛИТ, 2003. – 280с.
41. Understading Euler Angles [Електронний ресурс] - <http://www.chrobotics.com/library/understanding-euler-angles>
42. 3DR ArduCopter Quad-C [Електронний ресурс] - <https://www.robotshop.com/media/files/pdf/assembly-instructions-kt-ac3dr-01.pdf>
43. <http://store.jdrones.com/AC2836_358_Motors_p/ac2836358.htm> - [Електронний ресурс].
44. [Електронний ресурс] - <http://www.engr.colostate.edu/~dga/mech324/Labs/Lab%2010/images/moment%20of%20inertia%20table.jpg>
45. Tomas Jiinec. // [Електронний ресурс] - <https://pure.ltu.se/ws/files/33849977/LTU-EX-2011-33770150.pdf>
46. IMU sensor comparison [Електронний ресурс] - <http://snowcap.fi/blog/2012/01/04/imu-sensor-comparison/>
47. Gibiansky, Andrew. Quadcopter Dynamics and Simulation. [Електронний ресурс] - <http://andrew.gibiansky.com/downloads/pdf/Quadcopter%20Dynamics,%20Simulation,%20and%20Control.pdf>
48. Beard, Randal W. “Quadrotor dynamics and control.” Brigham Young University (2008).
49. Bouadi, H., M. Bouchoucha, and M. Tadjine. “Sliding mode control based on backstepping approach for an UAV type-quadrotor.” World Academy of Science, Engineering and Technology 26 (2007): 22-27.
50. Cai, Guowei, Ben M. Chen, and Tong H. Lee. Unmanned Rotorcraft Systems. New York: Springer, 2011
51. Carrillo, L. R. G., López, A. E. D., Lozano, R. and Pégard, C. (2013). Quad Rotorcraft Control, Advances in Industrial Control, Springer-Verlag, London.
52. DiCesare, Antonio. Design Optimization of a Quad-Rotor Capable of Autonomous Flight. Diss. WORCESTER POLYTECHNIC INSTITUTE, 2008.
53. Hoffmann, Gabriel M. Autonomy for sensor-rich vehicles: Interaction between sensing and control actions. ProQuest, 2008.
54. Koks, Don and Defence Science and Technology Organisation (Australia). Systems Sciences Laboratory. Using rotations to build aerospace coordinate systems. Edinburgh, S. Aust 2005
55. Mian, Ashfaq Ahmad, and Dao-bo Wang. “Dynamic modeling and nonlinear control strategy for an underactuated quad rotor rotorcraft.” Journal of Zhejiang University Science A 9.4 (2008): 539-545.
56. Min, Byung-Cheol, et al. “Development of a micro quad-rotor UAV for monitoring an Indoor environment.” Advances in Robotics. Springer Berlin Heidelberg, 2009. 262-271.
57. R. N. Jazar, Theory of Applied Robotics: Kinematics, Dynamics, and Control, Springer, New York, NY, USA, 2nd edition, 2010.
58. Tayebi, A.; McGilvray, S., “Attitude stabilization of a four-rotor aerial robot,” Decision and Control, 2004. CDC. 43rd IEEE Conference on , vol.2, no., pp.1216,1221 Vol.2, 14-17 Dec. 2004
59. Tayebi, Abdelhamid, and Stephen McGilvray. “Attitude stabilization of a VTOL quadrotor aircraft.” Control Systems Technology, IEEE Transactions on 14.3 (2006): 562-571.
60. <https://www.u-blox.com/sites/default/files/products/documents/LEA-6_ProductSummary_%28GPS.G6-HW-09002%29.pdf?utm_source=en%2Fimages%2Fdownloads%2FProduct_Docs%2FLEA-6_ProductSummary_%28GPS.G6-HW-09002%29.pdf> – [Електронний ресурс].
61. <https://aerocontent.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf> – [Електронний ресурс].
62. <https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf> – [Електронний ресурс].
63. <http://www.te.com/usa-en/products/brands/meas.html> – [Електронний ресурс].
64. <https://www.digikey.com/products/en/sensors-transducers/optical-sensors-photointerrupters-slot-type-transistor-output/548?k=46-1020-ND> – [Електронний ресурс].
65. http://www.utopiamechanicus.com/article/arduino-analog-pins-photointerruptor-or-slotted-optical/ – [Електронний ресурс].
66. http://fritzing.org/projects/by-tag/photointerrupter/ – [Електронний ресурс].
67. <http://robotic-controls.com/learn/projects/dji-esc-and-brushless-motor> – [Електронний ресурс].
68. Johan Wiberg, “Controlling a Brushless DC Motor in a Shift-by-Wire System” Master’s thesis performed in Vehicular Systems Reg nr: LiTH-ISY-EX-3517-2003
69. Petko Petkov, Tsonyo Slavov, “Stochastic Modeling of MEMS Inertial Sensors” -// BULGARIAN ACADEMY OF SCIENCES, CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume 10, No 2. – Sofia. 2010.
70. Tomáš Jiřinec, “Stabilization and Control of an Unmanned Quadcop”, MASTER'S THESIS. - Prague, May 30, 2011.
71. Improving the Beginner’s PID [Електронний ресурс] <http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>

**ДОДАТОК А**

**Вихідний код Python для здійснення відеопотоку**

import cv2

import numpy as np

import os

import time

def main():  
  
    capWebcam = cv2.VideoCapture(0) #оголошення захвату відео з об'єкта і зв’язок #з веб-камерою  
  
    print "defaultresolution = " + str(capWebcam.get(cv2.CAP\_PROP\_FRAME\_WIDTH)) + "x" + str(capWebcam.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)) #показ оригінаольного #розширення камери  
  
    capWebcam.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 320.0) #трансформація розширення камери  
    capWebcam.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 240.0) #на меньше значення для #кращої швидкодії  
  
    if capWebcam.isOpened() == False: #перевірка підключення камери  
        print "error: cap Webcam not accessed successfully\n\n"  
        os.system("pause")  
        return  
    # endif  
    while cv2.waitKey(1) != 27 andcapWebcam.isOpened(): # програма працює

#до тих пір,

# поки не   
        blnFrameReadSuccessfully, imgOriginal = capWebcam.read() # нажата кнопка ESC #чи камера   
 # не роз’єднається  
        if not blnFrameReadSuccessfully or imgOriginal is None: #перевірка на #наявність кадру з камери  
            print "error: frame not read from webcam\n"  
                        os.system("pause")  
            break  
        # endif  
        cv2.namedWindow("imgOriginal", cv2.WINDOW\_AUTOSIZE)  #вивід картинки

#на екран        
        cv2.imshow("imgOriginal", imgOriginal)       # монітора  
    # endwhile  
    cv2.destroyAllWindows() #очищення пам’яті від картинки  
    return #знову на початок циклу  
#############################################################  
if\_\_name\_\_ == "\_\_main\_\_":  
    main()

**ДОДАТОК Б**

**Вихідний код Python реалізації алгоритму розпізнавання об’єкту за кольором**

import cv2  
import numpy as np  
import os  
#############################################################  
def main():  
    capWebcam = cv2.VideoCapture(0) #оголошення захвату відео з об'єкта і зв’язок #з веб-камерою  
    print "defaultresolution = " + str(capWebcam.get(cv2.CAP\_PROP\_FRAME\_WIDTH)) + "x" + str(capWebcam.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)) #показ оригінаольного #розширення камери  
    capWebcam.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 320.0) #трансформація #розширення камери  
    capWebcam.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 240.0) #на менше значення для #кращої швидкодії  
    if capWebcam.isOpened() == False: #перевірка підключення #камери  
        print "error: cap Webcam not accessed successfully\n\n"  
        os.system("pause")  
        return  
    # endif  
    while cv2.waitKey(1) != 27 andcapWebcam.isOpened(): #програма працює до

#тих пір, поки не   
        blnFrameReadSuccessfully, imgOriginal = capWebcam.read() #нажата кнопка ESC #чи камера не   
 #роз’єднається  
        if not blnFrameReadSuccessfully or imgOriginal is None: #перевірка на #наявність кадру з камери  
            print "error: frame not read from webcam\n"  
                        os.system("pause")  
            break  
        # endif  
        #трансформація картинки з BGR в HSV простір

imgHSV = cv2.cvtColor(imgOriginal, cv2.COLOR\_BGR2HSV)   
 #оголошення діапазону знаходження цікавлючого кольору на зображені  
        imgThreshLow = cv2.inRange(imgHSV, np.array([0, 170, 120]), np.array([20, 240, 255]))  
        imgThreshHigh = cv2.inRange(imgHSV, np.array([20, 70, 170]), np.array([40, 170, 255]))  
        imgThresh = cv2.add(imgThreshLow, imgThreshHigh)  
 #створення маски зображення цікавлючого кольору  
        imgThresh = cv2.cvtColor(imgThresh, cv2.COLOR\_GRAY2RGB)  
 #зглажування об’єкта методом Гаусса  
        imgThresh = cv2.GaussianBlur(imgThresh, (5, 5), 2)

#морфологічна операція по розширенню об`єкта для заповнення випадкового простіру в об’єкті  
        imgThresh = cv2.dilate(imgThresh, np.ones((5,5),np.uint8))  
 #трансформація картинки в сірі тона  
        imgThresh = cv2.cvtColor(imgThresh, cv2.COLOR\_BGR2GRAY)  
 #бінарізація зображення  
        \_, imgThresh = cv2.threshold(imgThresh, 130, 255, cv2.THRESH\_BINARY)

#морфологічні операції для покращення цільності об’єкта на зображені  
        st1 = cv2.getStructuringElement(cv2.MORPH\_RECT, (21, 21), (10, 10))  
        st2 = cv2.getStructuringElement(cv2.MORPH\_RECT, (11, 11), (5, 5))  
        imgThresh = cv2.morphologyEx(imgThresh, cv2.MORPH\_CLOSE, st1)  
        imgThresh = cv2.morphologyEx(imgThresh, cv2.MORPH\_OPEN, st2)

#знаходження контурів об’єкта  
        img, contours,hierarchy = cv2.findContours(imgThresh, 2, 4)  
 #формування прямокутника (зеленого кольору) навколо об’єкта

try:  
        cnt = contours[0]

except Exception: continue

else:  
        a,b,w,h = cv2.boundingRect(cnt)  
        cv2.rectangle(imgOriginal,(a,b),(a+w,b+h),(0,255,0),2)

#знаходження центра об’єкта  
 x = a+w/2

y = b+h/2

#вивід картинки на екран монітора  
        cv2.namedWindow("imgOriginal", cv2.WINDOW\_AUTOSIZE)   
        cv2.namedWindow("imgThresh", cv2.WINDOW\_AUTOSIZE)  
        cv2.imshow("imgOriginal", imgOriginal)  
        cv2.imshow("imgThresh", imgThresh)  
    # endwhile  
    cv2.destroyAllWindows() #очищення пам’яті від картинки  
    return #знову на початок циклу  
#############################################################  
if\_\_name\_\_ == "\_\_main\_\_":  
    main()

**ДОДАТОК В**

**Вихідний код С++ реалізації ідентифікації об’єкта за особливими точками**

#include <iostream>

#include "opencv2/opencv.hpp"

#include "opencv2/core/core.hpp"

#include "opencv2/nonfree/features2d.hpp"

#include <vector>

using namespace std;

using namespace cv;

void readme(string &message)

{ cout << message << endl; }

int main( int argc, char\*\* argv )

{

VideoCapture cap(1);

if(!cap.isOpened()) // Проверка корректности отработки

{

string message = "Please check camera";

readme(message);

return -1;

}

Mat img\_object = imread( argv[1], CV\_LOAD\_IMAGE\_GRAYSCALE );

for(;;)

{

Mat frame;

cap >> frame; // Отримати черговий фрейм з камери

Mat img\_scene = frame;

if( !img\_object.data || !img\_scene.data ) // Перевірка наявності інформації в матриці зображення {

string message = " Error of reading the frame";

readme(message);

}

int minHessian = 400;

SurfFeatureDetector detector( minHessian );

std::vector<KeyPoint> keypoints\_object, keypoints\_scene;

detector.detect( img\_object, keypoints\_object );

detector.detect( img\_scene, keypoints\_scene );

SurfDescriptorExtractor extractor;

Mat descriptors\_object, descriptors\_scene;

extractor.compute( img\_object, keypoints\_object, descriptors\_object );

extractor.compute( img\_scene, keypoints\_scene, descriptors\_scene );

FlannBasedMatcher matcher;

vector< DMatch > matches;

matcher.match( descriptors\_object, descriptors\_scene, matches );

double max\_dist = 0; double min\_dist = 100;

//-- Обчислення максимального і мінімального відстані серед всіх дескрипторів в просторі ознак

for( int i = 0; i < descriptors\_object.rows; i++ )

{

double dist = matches[i].distance;

if( dist < min\_dist ) min\_dist = dist;

if( dist > max\_dist ) max\_dist = dist;

}

printf("-- Max dist : %f \n", max\_dist );

printf("-- Min dist : %f \n", min\_dist );

//-- Відібрати тільки хороші матчі, відстань менше ніж 3 \* min\_dist

vector< DMatch > good\_matches;

for( int i = 0; i < descriptors\_object.rows; i++ )

{

if( matches[i].distance < 3 \* min\_dist )

{

good\_matches.push\_back( matches[i]);

}

}

Mat img\_matches;

//-- Намалювати хороші матчі

drawMatches( img\_object, keypoints\_object, img\_scene, keypoints\_scene,

good\_matches, img\_matches, Scalar::all(-1), Scalar::all(-1),

vector<char>(), DrawMatchesFlags::NOT\_DRAW\_SINGLE\_POINTS );

//-- Локалізація об'єктів

vector<Point2f> obj;

vector<Point2f> scene;

for( int i = 0; i < good\_matches.size(); i++ )

{

obj.push\_back( keypoints\_object[ good\_matches[i].queryIdx ].pt);

scene.push\_back( keypoints\_scene[ good\_matches[i].trainIdx ].pt);

}

Mat H = findHomography( obj, scene, CV\_RANSAC );

//-- Отримати "кути" зображення з цільовим об'єктом

std::vector<Point2f> obj\_corners(4);

obj\_corners[0] = cvPoint(0,0); obj\_corners[1] = cvPoint( img\_object.cols, 0 );

obj\_corners[2] = cvPoint( img\_object.cols, img\_object.rows ); obj\_corners[3] = cvPoint( 0, img\_object.rows );

std::vector<Point2f> scene\_corners(4);

//-- Відобразити кути цільового об'єкта, використовуючи знайдене перетворення, на сцену

perspectiveTransform( obj\_corners, scene\_corners, H);

//-- З’єднати відображені кути

line( img\_matches, scene\_corners[0] + Point2f( img\_object.cols, 0), scene\_corners[1] + Point2f( img\_object.cols, 0), Scalar(0, 255, 0), 4 );

line( img\_matches, scene\_corners[1] + Point2f( img\_object.cols, 0), scene\_corners[2] + Point2f( img\_object.cols, 0), Scalar( 0, 255, 0), 4 );

line( img\_matches, scene\_corners[2] + Point2f( img\_object.cols, 0), scene\_corners[3] + Point2f( img\_object.cols, 0), Scalar( 0, 255, 0), 4 );

line( img\_matches, scene\_corners[3] + Point2f( img\_object.cols, 0), scene\_corners[0] + Point2f( img\_object.cols, 0), Scalar( 0, 255, 0), 4 );

//-- Show detected matches

imshow( "Good Matches & Object detection", img\_matches );

if(waitKey(30) >= 0) break;

}

} //-- Кінець основного циклу

**ДОДАТОК Г**

**Скрипт Matlab для аналізу даних переданої функції двигуна**

% This script processes .mat files comprised of ArduPilot Mavlink messages

% converted from .tlog files to .mat files in Mission Planner. This script

% will model the accelermoter, gyroscope, and GPS sensors

%% Initialization

clear all;

close all;

clc;

% Add Paths

addpath logs

%% Load log

load MotorRPM;

%% Extract data

t\_start = 300;

t\_end = 780;

time = MotorRPM(t\_start:t\_end,1)/1e3;

PWM = MotorRPM(t\_start:t\_end,3);

RPM = MotorRPM(t\_start:t\_end,5)/2; % 2 blades/propellor

% Remove RPM < 2500 (sensor errors)

RPMinds = RPM >= 2500;

time = time(RPMinds);

PWM = PWM(RPMinds);

RPM = RPM(RPMinds);

%% Simulate System

tau = 0.15;

DC = 6.5;

s = tf('s');

G = DC/(tau\*s+1);

dt = 0.01;

t = time(1):dt:time(end);

for(i = 1:size(t,2))

if (t(i) > 4.5 && t(i) <= 6.5)

PWM\_sim(i) = 1000;

end

if (t(i) > 6.5 && t(i) <= 8.5)

PWM\_sim(i) = 1425;

end

if (t(i) > 8.5 && t(i) <= 10.5)

PWM\_sim(i) = 1475;

end

if (t(i) > 10.5 && t(i) <= 12.5)

PWM\_sim(i) = 1450;

end

if (t(i) > 12.5 && t(i) <= 14.5)

PWM\_sim(i) = 1390;

end

if (t(i) > 14.5 && t(i) <= 16.5)

PWM\_sim(i) = 1425;

end

if (t(i) > 16.5 && t(i) <= 18.5)

PWM\_sim(i) = 1480;

end

if (t(i) > 18.5 && t(i) <= 20.5)

PWM\_sim(i) = 1425;

end

if (t(i) > 20.5 &&t(i) <= 22.5)

PWM\_sim(i) = 1375;

end

if (t(i) > 22.5)

PWM\_sim(i) = 1000;

end

end

RPM\_sim = lsim(G,(PWM\_sim),t);

%% Discretized model

% Using simulated PWM commands

sysD = c2d(G,dt,'zoh');

for(k = 1:size(t,2))

if k == 1

RPM\_est(k,1) = 0;

end

if k > 1

RPM\_est(k,1) = .9355\*RPM\_est(k-1,1) + .4192\*PWM\_sim(k-1);

end

end

% Using recorded PWM commands

RCPer = median(diff(time));

sysD = c2d(G,RCPer,'zoh');

for(k = 1:size(time,1))

if k == 1

RPM\_est2(k,1) = 0;

end

if k > 1

RPM\_est2(k,1) = .8133\*RPM\_est2(k-1,1) + 1.214\*PWM(k-1);

end

end

RPM\_est2 = RPM\_est2-5525;

%% Plot

% figure();

% plot(time,PWM);

% hold on;

% plot(t,PWM\_sim,'r');

figure();

plot(time,RPM,'r');

hold on;

% plot(t,RPM\_sim-5525,'b');

% plot(t,RPM\_est-5525,'g');

plot(time,RPM\_est2,'k');

axis([8.5,22,3300,4200])

str = sprintf('Simulated vs Actual Motor Dynamics (DC = %.2f Tc = %.2f)',DC,tau);

title(str);

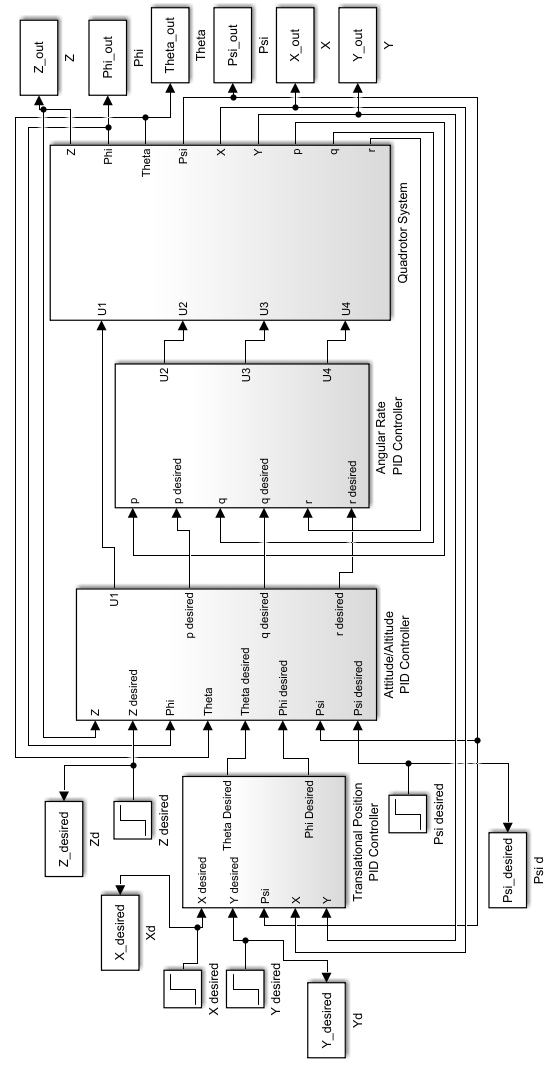
xlabel('Time (s)');

ylabel('RPM');

legend('Actual','Simulated');

**ДОДАТОК Г**

**Блок-схема взаємозв’язоку контролерів із системою керування квадрокоптера**



**ДОДАТОК Д**

**Вихідний код Matlab застосування фільтру Калмана для системи технічного зору**

clear all;

close all;

clc;

%% define main variables

dt = 1; %наша частота дискретизації (для прикладу роботи алгоритму = 1)

u = .01; % величина прискорення, береться від ІНС; для прикладу роботи алгоритму було взято 0.01

HexAccel\_noise\_mag = .1; % технологічнийшум: мінливість в тому, як швидко орієнтир прискорюється

tkn\_x = 1; % вимірювальний шум у горизонтальному напрямку (осі x).

tkn\_y = 1; % вимірювальний шум у горизонтальному напрямку (осі y).

Ez = [tkn\_x 0; 0 tkn\_y];

Ex = [dt^4/4 0 dt^3/2 0; ...

0 dt^4/4 0 dt^3/2; ...

dt^3/2 0 dt^2 0; ...

0 dt^3/2 0 dt^2].\*HexAccel\_noise\_mag^2; % Ex перетворює технологічний шум (stdv) в матрицю коваріації

P = Ex; % оцінка вихідної дисперсії положення орієнтира (матриця коваріації)

%% рівняння оновлення в 2-D! (Коефіцієнт матриць): модель на основі фізики, для якої ми очікуємо, що орієнтир буде [перехідний стан (стан + швидкість)] + [контроль входу (прискорення)]

A = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1]; %оновлення матриці стану

B = [(dt^2/2); (dt^2/2); dt; dt];

C = [1 0 0 0; 0 1 0 0]; % це наша функція вимірювання C, яку ми застосовуємо до оцінки стану Q, щоб отримати наше очікування наступного / нового вимірювання

%% ініціювати змінні результату

% Ініціалізувати для швидкості

Q\_loc\_meas = []; % значення координат орієнтира від СТЗ

%% ініціалізувати оцінювальні змінні

Q\_loc\_estimate = []; % позиція

vel\_estimate = []; % швидкість

P\_estimate = P;

predic\_state = [];

predic\_var = [];

webcamlist

cam = webcam('Logitech')

cam.Resolution = '320x240';

videoFrame = snapshot(cam);

frameSize = size(videoFrame);

videoPlayer = vision.VideoPlayer('Position', [100 100 [frameSize(2), frameSize(1)]+30]);

runLoop = true;

numPts = 0;

time = 0;

Center\_null = [1 1];

Radii\_null = 1;

Metric = 1;

detected\_someth = false; %чи було знайдено будь-який схожий об’єкт інтересу?

detected\_obj = false; %чи було знайдено наш об’єкт інтересу?

detected\_obj\_ini = true; %чи було знайдено наш об’єкт інтересу раніше \*true - NO?

while runLoop

videoFrame = snapshot(cam);

sceneImage = rgb2gray(videoFrame);

radiusRange = [10 30];

[centers,radii, metric] = imfindcircles(sceneImage,radiusRange, 'ObjectPolarity','bright'); %для тестування використовуємо простий алгоритм ідентифікації орієнтира – знаходження кола

try

Center = centers(1,:);

Radii = radii(1,:);

Metric = metric(1,:);

detected\_someth = true;

catch

detected\_obj = false;

end

if detected\_someth == true

if Metric > 0.4

cam\_Xc = Center(1);

cam\_Yc = Center(2);

detected\_obj = true;

Radii\_c = Radii;

else

detected\_obj = false;

end

else

detected\_obj = false;

end

if detected\_obj == true

if detected\_obj\_ini == true

Q= [cam\_Xc; cam\_Yc; 0; 0]; % ініційований стан - він має чотири компоненти: [positionX; позиція Y; швидкість X; швидкість Y] орієнтира

Q\_estimate = Q; % оцінка первинного місця розташування, де знаходиться орієнтир (що ми оновлюємо)

time = 0;

detected\_obj\_ini = false;

end

Q\_loc\_meas = [cam\_Xc; cam\_Yc];

end

if detected\_obj == false

time = time + 2;

if time >= 60

detected\_obj\_ini = true;

detected\_obj = false;

end

end

if detected\_obj\_ini == false

% Прогнозування наступного стану орієнтиру з останнім станом і передбаченим рухом.

Q\_estimate = A \* Q\_estimate + B \* u;

%predic\_state = [predic\_state; Q\_estimate(1)] ;

% передбачити наступну коваріацію

P = A \* P \* A' + Ex;

%predic\_var = [predic\_var; P] ;

% Kalman Gain

K = P\*C'\*inv(C\*P\*C'+Ez);

% Оновити оцінку стану.

if detected\_obj == true

Q\_estimate = Q\_estimate + K \* (Q\_loc\_meas - C \* Q\_estimate);

time = 0;

sceneImage = insertShape(sceneImage, 'circle', [[cam\_Xc cam\_Yc] Radii], 'LineWidth', 3);

else

cam\_Xc = Q\_estimate(1);

cam\_Yc = Q\_estimate(2);

sceneImage = insertShape(sceneImage, 'circle', [[cam\_Xc cam\_Yc] Radii], 'LineWidth', 3, 'Color', 'g');

end

% Оновити оцінку коваріації.

P = (eye(4)-K\*C)\*P;

else

cam\_Xc = -1;

cam\_Yc = -1;

sceneImage = insertShape(sceneImage, 'circle', [Center\_null Radii\_null], 'LineWidth', 3);

end

step(videoPlayer, sceneImage);

runLoop = isOpen(videoPlayer);

end

**ДОДАТОК Е**

**Вихідний код Matlab реалізації алгоритму побудови точок траєкторії посадки квадрокоптера**

function [ waypoints , psi\_dir\_end ] = waypoints\_calc( x\_coord, y\_coord, altitude )

if x\_coord < 0

pos\_x = -1;

x\_coord = abs(x\_coord);

else

pos\_x = 1;

end

if y\_coord < 0

pos\_y = -1;

y\_coord = abs(y\_coord);

else

pos\_y = 1;

end

altitude = altitude - 2; %????? ? 2 ????? ??? ??????????

psi\_dir = atan(y\_coord/x\_coord)\*pos\_y;

if pos\_x == -1

psi\_dir = pi - psi\_dir;

end

%psi\_dir\_end = psi\_dir \* (180/pi);

psi\_dir\_end = psi\_dir;

xy\_coord = sqrt(x\_coord^2 + y\_coord^2);

%Est = 4;

%E = ( pi \* xy\_coord \* altitude + ( xy\_coord - altitude )^2 ) / ( xy\_coord + altitude);

%St = E / Est;

%phi\_counter = pi/(2\*St);

phi\_counter = pi/6;

phi\_polar\_VS = 0:phi\_counter:pi/2;

p\_polar\_VS = xy\_coord\*altitude./sqrt((xy\_coord^2) \* (sin(phi\_polar\_VS)).^2 + (altitude^2) \* (cos(phi\_polar\_VS)).^2 );

z\_waypoints\_VS = p\_polar\_VS .\* sin(phi\_polar\_VS);

xy\_waypoints\_VS = p\_polar\_VS .\* cos(phi\_polar\_VS);

y\_waypoints\_VS = xy\_waypoints\_VS.\* ( y\_coord / xy\_coord);

x\_waypoints\_VS = xy\_waypoints\_VS.\* ( x\_coord / xy\_coord);

n = length(xy\_waypoints\_VS);

waypoints\_T = zeros(n,3);

waypoints\_T(1:n,1,:) = x\_waypoints\_VS;

waypoints\_T(1:n,2,:) = y\_waypoints\_VS;

waypoints\_T(1:n,3,:) = z\_waypoints\_VS;

waypoints\_T(1,:,:) = [ x\_coord y\_coord 0 ];

waypoints\_T(n,:,:) = [ 0 0 altitude ];

waypoints\_T(1:n,1,:) = waypoints\_T(1:n,1,:) .\* pos\_x;

waypoints\_T(1:n,2,:) = waypoints\_T(1:n,2,:) .\* pos\_y;

%waypoints\_T(1:n,2,:) = waypoints\_T(1:n,2,:) .\* (-1);

waypoints\_T(1:n,3,:) = waypoints\_T(1:n,3,:) - 30;

waypoints\_T(1:n,3,:) = waypoints\_T(1:n,3,:).\* (-1);

waypoints = waypoints\_T;

for i = 1:n

waypoints(i,:,:) = waypoints\_T(n+1-i, :, :);

waypoints(i,3,:) = waypoints(i,3,:) - 2;

end

%waypoints(1:n,3,:) = waypoints(1:n,3,:).\* (-1);

end