НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“____”_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності *122 – “Комп’ютерні науки”*

на тему: **Система відстеження та утримування об’єкта на кадрі в заданих координатах**

Виконав (-ла):
студент (-ка) IV курсу, групи ТР-02

ХІЛОБОК Олександр Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: *доцент каф. цифрових технологій в енергетиці*
доцент, к.т.н. Олександр КРЯЧОК _____
(посада, вчене звання ,науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент: _____
_____
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Н. контроль: *асистент Антон ПАСІЧНЮК* _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський).
Спеціальність 122 – “Комп’ютерні науки”.
Освітньо–професійна програма “Цифрові технології в енергетиці”.

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
“___”_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

ХІЛОБКУ Олександру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи: **Система відстеження та утримування об’єкта на кадрі в заданих координатах**

Керівник роботи: Крячок Олександр Степанович к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “___” _____ 202__ р. № _____

2. Термін подання роботи студентом 07.06.2024 року

3. Вихідні дані до роботи: мова програмування Python, MicroPython, середовище розробки PyCharm, бібліотека для реалізації комп’ютерного зору OpenCV.

4. Перелік питань, які необхідно розробити:

1) Визначити найкращий метод відслідковування об’єкту.

2) Розробити програму для відслідковування в реальному часі.

3) Розробити механізм для повороту камери за допомогою мікроконтролера.

4) Програмно синхронізувати роботу контролера з основною програмою.

5. Орієнтовний перелік графічного (ілюстративного) матеріалу: приклади роботи програми відслідковування, таблиця порівняння трекерів, фотографії готового механізму та панелі керування системою, фотографії використаних компонентів.

6. Дата видачі завдання “15” вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи	19.09.2024	*виконано*
2.	Вивчення та аналіз задачі	17-20.04.24	*виконано*
3.	Розробка архітектури та загальної структури системи	21-25.04.24	*виконано*
4.	Програмна реалізація системи	25.04-02.05.24	*виконано*
5.	Розробка механізму	03-07.05.24	*виконано*
6.	Оформлення пояснювальної записки	08-12.05.24	*виконано*
7.	Захист програмного продукту	13-17.05.24	*виконано*
8.	Передзахист	03-06.06.24	*виконано*
9.	Подання готової роботи на кафедру	07.06.2024	*виконано*
10.	Захист	17-21.06.2024	

Студент _____ **Олександр ХІЛОБОК**
(підпис) (ім’я, ПРІЗВИЩЕ)

Керівник _____ **Олександр КРЯЧОК**
(підпис) (ім’я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 42 сторінках, містить 9 ілюстрацій, 2 таблиці, 1 додаток, 13 джерел в переліку посилань.

Робота присвячена розробці системи, що відслідковує обраний об’єкт та утримує його в полі зору камери. Система складається з програмної та апаратної частини: програмного продукту та механізму повороту.

Тема обумовлює свою актуальність швидким розвитком технологій комп’ютерного зору, потужності сучасних процесорів та потребами багатьох сфер в інтеграції сучасних технологій комп’ютерного зору.

Для вирішення проблеми була використана мова Python та популярний модуль OpenCV Python що спрощує розробку систем комп’ютерного зору, також був використаний мікроконтролер Raspberry Pi Pico і мова програмування MicroPython для обміну даними, контролю сервоприводами та повороту камери.

В результаті роботи було протестовано та порівняно роботу доступних в модулі OpenCV методів відстежування, розроблено систему відстеження об’єкту та утримування його на кадрі в заданих координатах. Система працює справно та може відстежувати об’єкти, які швидко рухаються та повільно змінюють свій колір і форму.

Ключові слова: трекінг, відстежування, OpenCV, комп’ютерний зір, камера, мікроконтролер.

ABSTRACT

The thesis consists of 42 pages, contains 9 illustrations, 2 tables, 1 appendix, 13 sources in the list of references.

The work is devoted to the development of a system that tracks the selected object and keeps it in the field of view of the camera. The system consists of a software and hardware part: a software product and a turning mechanism.

The subject is relevant due to the rapid development of computer vision technologies, the power of modern processors, and the needs of many fields in the integration of modern computer vision technologies.

To solve the problem, the Python language and the popular OpenCV Python module, which simplifies the development of computer vision systems, were used. The Raspberry Pi Pico microcontroller and the MicroPython programming language were also used for data exchange, servo control and camera rotation.

As a result of the work, the operation of the tracking methods available in the OpenCV module was tested and compared, a system for tracking the object and keeping it on the frame in the given coordinates was developed. The system works well and can track objects that move quickly and slowly change their color and shape.

Keywords: tracking, OpenCV, computer vision, camera, microcontroller.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE (integrated development environment) – Інтегроване середовище розробки.

Трекінг – це відстеження траєкторії руху деякого об’єкту.

Мікроконтролер – це спеціалізований програмований прилад, що призначений для використання у керуючих пристроях, системах передачі даних та системах керування технологічними процесами.

FPV – вид від першої особи, зазвичай використовується для позначення типу керування безпілотних апаратів.

БПЛА – безпілотний літаючий апарат.

CUDA – це архітектура, яка дозволяє використовувати графічний процесор для підвищення продуктивності паралельних обчислень. Вона являє собою набір інструментів та бібліотек для роботи з графічним процесором.

COM порт – послідовний порт передачі даних від однієї цифрової системи до іншої, в сучасних системах був витіснений USB портом, який також підтримує цю технологію.

ВСТУП

У сучасному світі автоматизація тих процесів, що раніше виконували люди набирає все більших обертів, іноді це відкриває нові можливості в різних галузях, де раніше швидкості роботи та реакції людини не вистачало, а в деяких місцях просто було небезпечно. Комп'ютерний зір – одна з галузей, яка дуже широко використовується при автоматизації різної роботи, як, наприклад: класифікація, підрахунок, спостерігання за об'єктом, який знаходиться в полі зору камери.

Метою даної дипломної роботи є дослідження та розробка системи, що тримає обраний об'єкт в полі зору камери при зміні його положення та форми в реальному часі.

Актуальність предметної області досліджень обумовлена зростанням потреб в автоматизації та розповсюдження технологій комп'ютерного зору і нейронних мереж, а також зростанням потреб України в розвитку оборонної промисловості, де продукт даного проекту може отримати своє продовження у вигляді системи спостережень за переміщенням військової техніки в реальному часі для БПЛА, а також системи наведення для недорогих БПЛА-камікадзе власного виробництва.

Предметом досліджень є методи відслідковування руху об'єкту на зображеннях та відео в реальному часі. Об'єктом дослідження є оцінка швидкодії та ефективності цих методів і розробка системи відслідковування на їх основі.

Метою даної дипломної роботи є розробка системи, яка утримує обраний об'єкт на кадрі в заданих координатах, при цьому система має надійно працювати навіть коли об'єкт змінює свої параметри, колір, форму та швидкість руху, розроблена система має вміти відслідковувати об'єкти, які рухаються з великою швидкістю, тобто мати достатню кутову швидкість повороту та обробляти нові зображення з достатньою частотою, щоб не втрачати такий об'єкт.

Для досягнення мети були поставлені такі задачі:

1. Аналіз доступних програмних засобів для реалізації систем комп'ютерного зору.

2. Розробка логіки роботи програми, основних алгоритмів.

3. Розробка системи трекінгу, відслідковування об’єктів на основі розглянутих програмних засобів.

4. Порівняння різних методів трекінгу для виявлення найбільш підходящого для реалізації поставленої мети.

5. Розробка механізму повороту камери та написання прошивки для мікроконтролеру що керує сервоприводами.

6. Налагодження синхронної роботи мікроконтролера з комп’ютером, що виконує обробку зображення

7. Тестування отриманої системи та оптимізація для збільшення швидкодії, виправлення помилок в роботі програм.

Структура роботи включає перелік умовних позначень, скорочень і термінів, вступ, 4 розділи основної частини, висновки, список використаних джерел, додаток. У розділі 1 описані основні протестовані методи та можливості використання. У розділі 2 розказано про програмні методи реалізації, мови програмування та використані модулі. У розділі 3 описаний вибір мікроконтролеру, та розробка механізму. У розділі 4 описані особливості програмної реалізації, програмна архітектура системи, опис роботи програми та її функціонал.

1 РОЗРОБКА СИСТЕМИ ДЛЯ ВІДСЛІДКОВУВАННЯ ОБ'ЄКТУ

Програмний застосунок, розроблений за темою дипломної роботи, має змогу відслідковувати зафіксований об'єкт за допомогою обраного методу. Метод відслідковування обирається при запуску програми для порівняння різних доступних методів на практиці. Для отримання зображення в реальному часі програма отримує доступ до однієї з доступних веб камер.

1.1 Потенційна сфера використання

Наразі основною сферою використання на яку орієнтується даний проект при розробці – це впровадження систем машинного зору в військовій сфері країни. Розробка власних систем дозволить Україні зробити великий крок вперед у розробці військових технологій для захисту держави. Країни союзники та недружні сторони вже давно мають власні системи штучного інтелекту та машинного зору, які впроваджені у військову сферу. Такі системи можуть як допомагати пілотам FPV дронів більш точно наводитись на військові цілі, так і мати повний контроль над БПЛА та дронами камікадзе, при цьому досить точно влучати у поставлені цілі, або слідкувати та збирати розвід дані в автоматичному режимі.

Розробка власної системи допоможе зберегти багато життів військових на фронті та забезпечити гарний поштовх для впровадження сучасних технологій в армію України на майбутнє.

1.2 Задачі для програмного забезпечення

Для досягнення поставлених цілей треба встановити чіткі задачі до розробки програмного забезпечення. Програма має відповідати наступним вимогам:

1. Обирати потрібний об’єкт. Для того щоб відслідковувати рух об’єкту потрібно спочатку його виділити, для цього програма має мати змогу змінювати розмір зони виділення.

2. Слідкувати за об’єктом. Після виділення потрібної зони програма має перемикатись в режим слідкування та надійно відслідковувати точну траєкторію руху об’єкту, навіть якщо він швидко рухається та змінює свій колір та форму, або змінюється освітлення, або розмір.

3. Мати змогу зняти виділення з об’єкту в будь який момент та обрати інший, змінюючи розмір. Наприклад, якщо програма втратила останній об’єкт через неякісне зображення

4. Мати змогу обирати різні методи відслідковування для порівняння швидкодії та якості.

1.3 Аналіз доступних методів відслідковування

Розглянемо деякі досить відомі методи трекінгу, які можна використати для відслідковування траєкторії руху об’єкту в даному проекті.

Будуть розглянуті наступні методи:

1. Boosting Tracker.
2. MIL (Multiple Instance Learning) Tracker.
3. KCF (Kernelized Correlation Filters) Tracker.
4. TLD (Tracking Learning Detection) Tracker.
5. Median Flow Tracker.
6. CSRT Tracker.
7. MOSSE Tracker.
8. GOTURN (Generic Object Tracking Using Regression Network) Tracker.

BOOSTLING – заснований на онлайн–версії алгоритму AdaBoost ’алгоритм збільшує ваги неправильно класифікованих об’єктів, що дозволяє слабкому класифікатору «фокусуватися» на їх виявленні. Оскільки класифікатор навчається

«онлайн», користувач задає фрейм, у якому знаходиться об’єкт відстеження. Цей об’єкт спочатку розглядається як позитивний результат виявлення, а об’єкти навколо нього розглядаються як фон. Отримавши новий кадр зображення, класифікатор оцінює навколишні пікселі виявлення з попереднього кадру, і нове положення об’єкта буде областю, де оцінка має максимальне значення. Плюси: об’єкт відстежується досить точно, навіть якщо алгоритм вже застарів. Мінуси: відносно низька швидкість, сильна сприйнятливість до шумів і перешкод, а також неможливість зупинити стеження при втраті об’єкта [7].

MIL – має той самий підхід, що й BOOSTING, однак замість вгадування, де знаходиться відстежуваний об’єкт у наступному кадрі, використовується підхід, у якому кілька потенційно позитивних об’єктів, які називаються «мішком», вибираються навколо позитивно визначеного об’єкта. Позитивний «мішок» містить принаймні один позитивний результат. Плюси: більш стійкий до шуму, показує досить хорошу точність. Мінуси: відносно низька швидкість і неможливість зупинити стеження при втраті об’єкта [7].

KCF – це комбінація двох алгоритмів: BOOSTING і MIL. Концепція методу полягає в тому, що набір зображень із «мішка», отриманих методом MIL, має багато областей, що перекриваються. Кореляційна фільтрація, застосована до цих ділянок, дозволяє з високою точністю відстежувати переміщення об’єкта і прогнозувати його подальше положення. Плюси: досить висока швидкість і точність, припиняє стеження при втраті відстежуваного об’єкта. Мінуси: неможливість продовжити стеження після втрати об’єкта [7].

TLD – дозволяє розкласти завдання відстеження об’єкта на три процеси: відстеження, навчання та виявлення. Трекер (на базі трекера MedianFlow) відстежує об’єкт, а детектор локалізує зовнішні ознаки і при необхідності коригує трекер. Навчальна частина оцінює помилки виявлення та запобігає їм у майбутньому шляхом розпізнавання пропущених або помилкових виявлень. Плюси: демонструє відносно хороші результати щодо стійкості до масштабування об’єктів і перекриття іншими об’єктами. Мінуси: досить непередбачувана

поведінка, є нестабільність виявлення та відстеження, постійна втрата об’єкта, відстеження схожих об’єктів замість обраного [7].

MEDIANFLOW – заснований на методі Лукаса-Канаде. Алгоритм відстежує в часі рух об’єкта в прямому і зворотному напрямках і оцінює похибку цих траєкторій, що дозволяє трекеру прогнозувати подальше положення об’єкта в реальному часі. Плюси: досить висока швидкість і точність стеження, якщо об’єкт не перекривається іншими об’єктами і швидкість його руху не надто висока. Алгоритм досить точно визначає втрату об’єкта. Мінуси: висока ймовірність втрати об’єкта при великій швидкості його руху [7].

CSRT – використовує карти просторової надійності для налаштування підтримки фільтра на частину виділеної області з кадру для відстеження, що дає можливість збільшити область пошуку та відстежувати непрямокутні об’єкти. Показники надійності відображають якість досліджуваних фільтрів по каналах і використовуються як вагові коефіцієнти для локалізації. Таким чином, використовуючи HoGs і Colornames як набори функцій, алгоритм працює відносно добре. Плюси: серед попередніх алгоритмів демонструє порівняно кращу точність, стійкість до перекриття іншими об’єктами. Мінуси: досить низька швидкість, нестабільна робота при втраті об’єкта [7].

MOSSE – заснований на розрахунку адаптивних кореляцій у просторі Фур’є. Фільтр мінімізує суму квадратів помилок між фактичним виходом кореляції та прогнозованим виходом кореляції. Цей трекер стійкий до змін освітлення, масштабу, пози та нежорстких деформацій об’єкта. Плюси: дуже висока швидкість відстеження, більш успішне продовження відстеження об’єкта, якщо він був втрачений. Мінуси: висока ймовірність продовження стеження, якщо об’єкт загубився і не з’явився в кадрі [7].

Алгоритм GOTURN Tracker є «офлайн» трекером, оскільки він в основному містить глибоку згорточну нейронну мережу. У мережу подається два зображення: «попереднє» і «поточне». На «попередньому» зображенні позиція об’єкта відома, тоді як на «поточному» зображенні положення об’єкта потрібно передбачити. Таким чином, обидва зображення проходять через згортову нейронну мережу,

результатом якої є набір із 4 точок, що представляють координати передбачуваної обмежувальної рамки, що містить об’єкт. Оскільки алгоритм заснований на використанні нейронної мережі, користувачеві необхідно завантажити та вказати файли моделі та ваги для подальшого відстеження об’єкта. Плюси: порівняно хороша стійкість до шуму та перешкод. Мінуси: точність відстеження об’єктів залежить від даних, на яких навчалася модель, а це означає, що алгоритм може погано відстежувати деякі об’єкти, вибрані користувачем. Втрачає об’єкт і переходить на інший, якщо швидкість першого занадто велика [7].

1.4 Порівняння швидкодії доступних методів

Для порівняння швидкодії трекерів всі трекери були запущені одночасно на одному відеоматеріалі з однаковою початковою областю відслідковування. (рисунок 1.1)

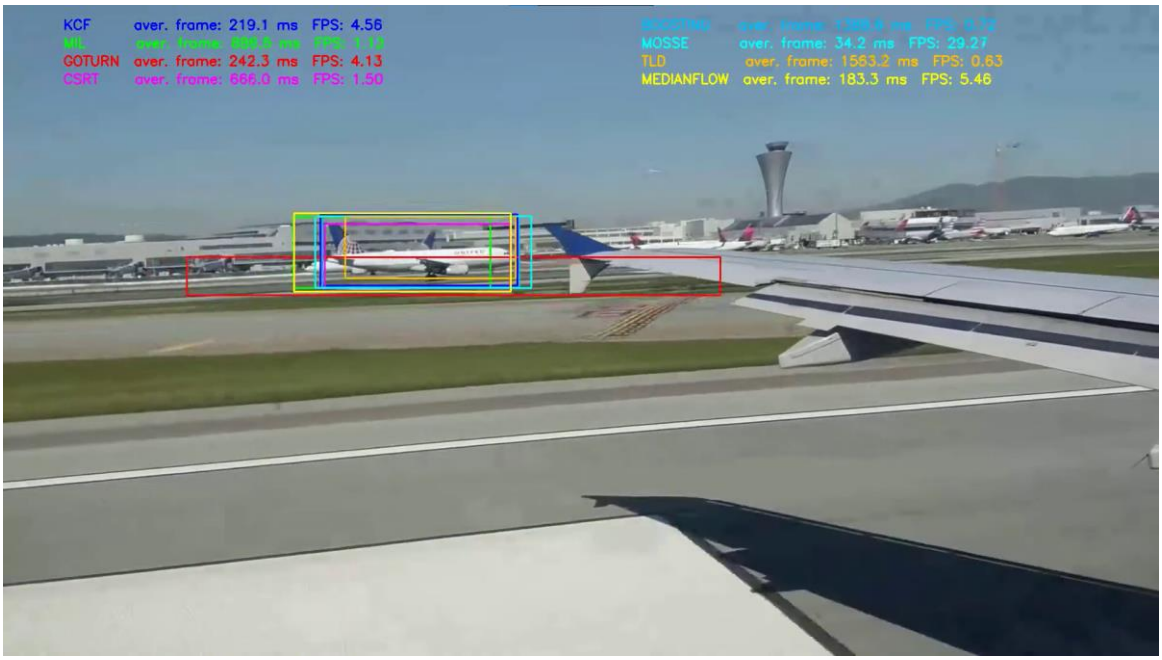

Рисунок 1.1 – Тестування всіх наявних трекерів

Отримані дані зі швидкодії трекерів були записані в таблицю 1.1:

Таблиця 1.1 – Порівняння швидкодії трекерів

Назва	Час на кадр (мс)	Кадрів в секунду
KCF	217	4.6
MIL	885	1.13
GOTURN	241	4.15
CSRT	675	1.48
BOOSTLING	1384	0.72
MOSSE	34	29.14
TLD	1540	0.65
MEDIANFLOW	184	5.43

Підсумовуючи значення з даної таблиці, можна сказати що найшвидшим трекером є MOSSE, який показав цілих 29 кадрів на секунду в порівнянні з іншими це дуже велика різниця.

Але в реальних умовах швидкодія не значить якість, тому реальне тестування на практиці показало, що трекер TLD, який є найповільнішим показав дуже хороше відпрацювання, але, нажаль, слідкування ламалося коли об’єкт починав змінювати форму або колір, тому перший по якості трекер було встановлено – CSRT. Об’єкт залишається в відслідковуванні навіть при зміні кольору та форми, а швидкість в декілька раз вище за попередній, але єдиний недолік в тому що він може втратити початкову ціль якщо вона зробить швидкий рух. Як виявилось цю проблему можливо частково вирішити збільшенням частоти кадрів.

Також потрібно відмітити, що якість трекінгу дуже сильно залежить від якості матеріалу для опрацювання, тобто відео, якщо якість камери буде поганою, трекер буде постійно втрачати об’єкт. На це дуже сильно впливає така характеристика камери, як витримка, вона визначає на який проміжок часу затвор камери залишається відкритим, тобто інтервал, в який на датчик зображення камери потрапляє світло, і чим менший цей час тим менший рух на зображенні встигне зробити об’єкт, а тому його зображення залишиться більш чітким,

витримку камери можна зменшувати до мінімуму, якщо в матриці вистачає світлочутливості, але від цього збільшується і ціна камери. Світлочутливі якісні матриці дуже дорогі.

1.5 Збільшення швидкості обробки зображення

Для зменшення часу на обробку кадру були протестовані декілька методів.

Основний метод, який показав найкращий результат в пришвидшенні обробки зображення це просте зменшення роздільної здатності зображення на вході в основну функцію оновлення трекеру. Тут все максимально логічно, менше зображення – менше окремих пікселів потрібно опрацювати програмі для завершення циклу оновлення, а тому і більша швидкість.

Якщо при звичайній роздільній здатності недорогої веб камери 1280x720 пікселів, програма на операційній системі Windows 10 та з центральним процесором Intel 10300H могла обробити близько 10-12 кадрів на секунду, то при роздільній здатності 640x360 частота оновлення досягала 22-24 кадрів, чого цілком достатньо для адекватної роботи програми та задовільних результатів, що відповідають меті.

При зображення 1920x1080 програма видавала близько 6-8 кадрів, що не дозволяло надійно відслідковувати об’єкти, які швидко рухаються та не задовольняло цілі оптимізації.

Інший спосіб пришвидшити роботу програми це використовувати чорно–біле зображення. При обробці зображення програма аналізує кожен піксель по трьом каналам: червоний, зелений, синій, де кожен канал може мати значення від 0 до 255, що відповідає сучасним стандартам зображення, 24 біта на кожен піксель, при цьому є близько 16 мільйонів кольорів які він може мати. Але якщо пропустити зображення через функцію, яка робить його чорно–білим, на виході отримаємо зображення, де на кожен піксель існує лише один канал – це градація чорного від 0

до 255, таким чином, в теорії, швидкість роботи програми має збільшитись в декілька разів [8].

Але на практиці такого значного результату отримано не було. Швидкість обробки зображення справді дещо збільшилась, тепер цикл міг проходити близько 24-28 разів на секунду, а просадки та мікро–зависання майже зникли, але в плату за такий незначний результат довелось дуже знизити якість трекінгу, адже тепер, замість 24 біт на піксель залишилось тільки 8, тобто було втрачено дуже багато інформації, яка допомагала більш точно відслідковувати траєкторію об’єкта.

Тому довелось відмовитись від даного методу прискорення, заради якості відслідковування об’єкту в кадрі.

Останнім методом є використання графічного прискорювача Nvidia з підтримкою технологій CUDA. Бібліотека OpenCV Python є відкритою та може підтримувати використання Nvidia CUDA, але нажаль в стандартному розповсюджуваному модулі немає такої можливості, для цього потрібно самому завантажити вихідний код бібліотеки та скомпілювати його з ввімкненою підтримкою даних технологій за допомогою програми CMake. Використання графічного прискорювача значно пришвидшує роботу програми, але це не можна назвати оптимізацією, до того ж обладнання Nvidia, що підтримує сучасні версії CUDA досить дороге, та споживає багато енергії, тому в проекті не буде використовуватись даний метод.

2 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі описані ключові засоби для розробки програмного забезпечення, що виконує функції відслідковування траєкторії руху об’єкту, також описана апаратна частина проекту, а саме опис обраного мікроконтролеру для керування сервоприводами, порівняння з доступними аналогами та засоби розробки прошивки для нього.

2.1 Мова програмування Python

Python є одною за найбільш розповсюджених з високорівневих мов програмування, її простота і наявність великої кількості ліцензійованих та користувацьких модулів, бібліотек робить її найкращою для розробки систем штучного інтелекту і комп’ютерного зору. Взагалом Python використовується для бекенду, швидкої розробки програм, а також як засіб поєднування наявних компонентів [1].

З основних переваг можна ще виділити:

1. Чистий синтаксис (для виділення блоків використовуються відступи) – має простий зрозумілий синтаксис, що дуже зручно для відладки та розробки, також це сприяє швидкому розумінню нових модулів або чужого коду.

2. Переносність програм – Python програми можна запускати на інших комп’ютерах бея будь яких проблем асе що потрібно це встановити сам Python та підвантажити необхідні модулі.

3. Стандартний дистрибутив має велику кількість корисних модулів – дуже велика кількість вбудованих модулів для роботи з функціоналом операційної системи, створення візуального інтерфейсу, роботи з мережею та інше.;

4. Зручна для розв’язання математичних проблем – простий синтаксис дозволяє швидко набирати складні математичні вирази, також вбудовані модулі мають багато корисних математичних функцій;

5. Кросплатформеність – дозволяє легко запускати один і той же код не просто на іншому комп’ютері, а ще й з підтримкою інших операційних систем;

Для програмної частини що відповідає за роботу трекеру була обрана саме мова Python, вона відповідає всім потребам в можливостях швидкої реалізації та швидкості роботи написаної програми. Також має велику підтримку зі сторони користувачів, постійні оновлення самого інтерпрітатору та доступних модулів.

Масове використання даної мови для створення систем машинного зору дуже спрощує подальшу розробку, так як багато потрібних методів вже були написані у вигляді завантажуваних модулів, таких як OpenCV.

2.2 Мова програмування MicroPython

MicroPython – це програмна реалізація мови програмування, яка багато у чому сумісна з Python 3, написана на мові C і оптимізована для роботи на мікроконтролерах [2].

Із всіх мов які підтримують роботу з мікроконтролерами була обрана саме MicroPython, через простоту використання, можливість швидкої розробки, високорівневість і велику кількість доступних модулів, які полегшують задачу розробки програмного забезпечення для мікроконтролеру.

MicroPython включає крос–компілятор, який генерує байт-код MicroPython (розширення файлу .mpy). Код Python може бути скомпільований у байт–код або безпосередньо на мікроконтролері, або його можна попередньо скомпілювати в іншому місці.

Прошивку MicroPython можна створити без компілятора, залишивши лише віртуальну машину, яка може запускати попередньо скомпільовані програми.

Також мова MicroPython, не зважаючи на свою високорівневість, включає в себе деякі модулі, які надають програмісту доступ до обладнання низького рівня.

2.3 Бібліотеки та модулі Python

Базою для реалізації програми трекінгу в даному проекті став загальнодоступний модуль з відкритим кодом OpenCV Python. Даний модуль був розроблений для створення простих та комплексних проектів у сфері машинного зору та нейронних мереж, широко використовується в мовах C, C++, Java, Python. Основним його призначенням є розробка програм що виконують обробку, аналіз та класифікацію зображення [4].

OpenCV наповнений алгоритмами, які допомагають у всьому, від розпізнавання об'єктів до відстеження руху об'єктів і навіть створення 3D-моделей. Він неймовірно популярний, має велику спільноту розробників по всьому світу.

OpenCV підтримує Windows, Linux, Android і MacOS. Завдяки ефективному використанню комп'ютерної обробки, це особливо добре для проектів, які повинні працювати в режимі реального часу. Завдяки постійному розвитку таких технологій, як CUDA та OpenCL, OpenCV є основою незліченних інноваційних програм у всьому світі [9].

Основна причина по якій був обраний саму цей модуль –це велика кількість методів для відслідковування траєкторії об'єкту

PySerial бібліотека для мови Python, яка призначена для створення зв'язку через послідовний COM порт комп'ютеру, в проекті вона буде використана для створення зв'язку з мікроконтроллером через USB вихід комп'ютеру і передачі команд між контролером та комп'ютером [5].

Також були використані такі бібліотеки для MicroPython:

1. micropython-servo, для спрощеного контролю над сервоприводами;

2. machine, вбудований в MicroPython модуль, який надає доступ до всіх пінів (gpio) мікроконтроллеру.

2.4 Інтегроване середовище розробки (IDE)

Для розробки програмного забезпечення всього проекту було використане середовище розробки PyCharm Community. PyCharm був розроблений компанією JetBrains – відомою міжнародною компанією, що спеціалізується на розробці інструментів для програмування різними мовами, як–от Java, Kotlin, C#, F#, C++, Ruby, Python, PHP, JavaScript і багатьох інших. Заснована раніше під ім’ям IntelliJ, вона сьогодні надає різноманітні засоби для командної роботи та підтримки розробки провідних мов програмування.

Можливості PyCharm:

1. Автодоповнення та інтелектуальні підказки. Це дає змогу швидше писати код, не витрачаючи час на пошук імен функцій, методів і змінних, а також скорочує ймовірність помилок під час їхнього виклику.

2. Статичний аналіз коду. Це допомагає поліпшити якість вашого коду і робить його більш надійним.

3. Налагодження та профілювання. Ви можете встановлювати точки зупинки, аналізувати значення змінних і покроково виконувати код, що допомагає зрозуміти, що відбувається в програмі на кожному етапі виконання. Інструменти профілювання дають змогу оптимізувати продуктивність вашого коду.

4. Рефакторинг та оптимізація. Це робить код більш читабельним, структурованим і підтримуваним.

5. Інтеграція з системами контролю версій, такими як Git. Це спрощує роботу з історією змін у вашому коді, комміти, розгалуження та вирішення конфліктів.

6. Віртуальне оточення PyCharm. Це ізольований простір для Python-додатків, який дає змогу мати свій власний набір залежностей, не впливаючи на інші проєкти. Це дає змогу розробляти й тестувати додатки в окремих, чистих середовищах, забезпечуючи надійність і запобігаючи конфліктам між залежностями різних проектів.

7. Підтримка фреймворків і технологій, таких як Django, Flask, SQLAlchemy, HTML, CSS, JavaScript та інші. Це дає змогу розробляти веб–додатки та інші проекти з використанням сучасних технологій [3].

PyCharm являється одним з найзручніших середовищ для розробки та відладки Python програм, його функціонал дозволяє дуже швидко писати код та виправляти помилки, а можливість завантажувати модифікації дає змогу навіть завантажувати програми на пряму до мікроконтролеру через PyCharm.

3. ПОБУДОВА МЕХАНІЗМУ ПОВОРОТУ

В даному розділу будуть описані етапи побудови механізму повороту камери, вибір мікроконтролеру для контролю сервоприводами та додаткові деталі, які знадобились.

3.1 Вибір мікроконтролеру для контролю сервоприводів

На сучасних маркетплейсах існує дуже великий вибір різних мікроконтроллерів, від невеликих процесорів, які програмуються через спеціальні програматори, до повноцінних плат де розпаяна пам’ять, зручний вивід всіх пінів та просте підключення до комп’ютеру через USB.

Однак, розглянемо декілька найпопулярніших варіантів, а саме: старий да дуже популярний Arduino Nano V3 на процесорі ATMega328P (рисунок 3.1)

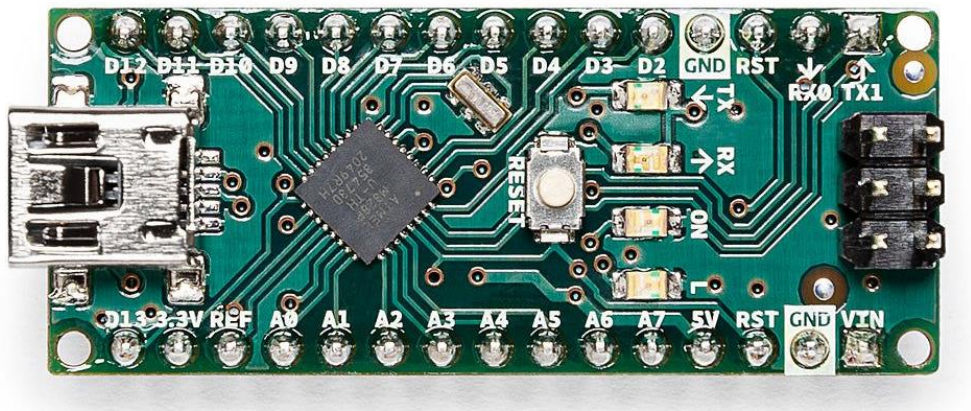

Рисунок 3.1 – Плата мікроконтролер Arduino Nano ATMega328P

Arduino – це платформа з відкритим кодом, яка допомагає розробникам схем створювати електронні проекти. Вона складається як з апаратного, так і з програмного забезпечення. Апаратне забезпечення Arduino – це програмована друкована плата, яка називається мікроконтролером. Програмне забезпечення – це

IDE (інтегроване середовище розробки), за допомогою якого розробники пишуть і завантажують код у мікроконтролер.

Ми можемо передати програму з набором інструкцій на плату Arduino, яка може виконувати як прості, так і складні завдання. Традиційні програмовані плати вимагають окремого обладнання для завантаження коду на плату. Але Arduino усуває потребу в апаратному забезпеченні; замість цього він використовує простий кабель USB для завантаження коду на плату.

Плата дозволяє розробникам створювати програму у спрощеній версії мови C++, що полегшує навчання та програмування [10].

Також розглянемо більш сучасний, набераючий популярність аналог Arduino – Raspberry Pi Pico 2020 року на процесорі RP2040 (рисунок 3.2)

Рисунок 3.2 – Плата мікроконтролер Raspberry Pi Pico RP2040

Обидві плати мають своє походження с зовсім різних часів розвитку технологій, в той час як Arduino Nano бере свій початок з 2008 року, коли тренд розробки власних проектів на невеликих мікроконтроллерах та автоматизації процесів в повсякденному житті тільки зароджувався, Pico навпаки була випущена в 2020 році як плата мікроконтроллер на заміну застарілим варіантам, сучасна Pico має перевагу майже по всім параметрам. Основні відмінності плат описані в таблиці 3.1

Таблиця3.1 – Порівняння характеристик мікроконтролерів

Microcontroller	ATMega328	RP2040 ARM Cortex M0+ dual–core
Частота	16MHz	133MHz
SRAM	2kB	264kB
Flash memory	32kB	2MB
EEPROM	1kB	–
GPIO pins	22	26
Analog in pins	8	3
PWM pins	6	16
I/O pins voltage	5V	3.3V
I/O pins current	40mA	~15mA
3.3V pin current	50mA	Max 300mA
Power voltage	7–12V	5V
Розміри	18x45mm	51x21mm
Мови	C alike	C, C++, MicroPython

Судячи з даних таблиці можна сказати що Raspberry Pi Pico є абсолютним переможцем, маючи більше пам’яті, потужніший процесор, більше підтримуваних мов програмування та меншу ціну на момент написання диплому. Тому була обрана саме ця плата мікроконтролер для контролю сервоприводами [6].

Також важливим фактором стала підтримка мови програмування MicroPython, яка дуже схожа з мовою Python, на якій написане основне програмне забезпечення – програма трекер, тому Raspberry Pi Pico є універсальним варіантом, а контролери попередніх поколінь не мають сенсу на даний момент.

3.2 Сервоприводи

Серводвигуни або «сервоприводи», як їх називають, – це електронні пристрої та поворотні або лінійні приводи, які з точністю обертають і штовхають механізм.

Сервоприводи в основному використовуються для кутового або лінійного положення та для певної швидкості та прискорення.

Компанії активно використовують сервомотори через те, наскільки вони компактні та потужні. Незважаючи на свій розмір, вони генерують досить багато енергії.

Більшість компаній, які використовують сервоприводи, є виробничими компаніями, яким вони потрібні для позиціонування поверхонь керування та обертання об’єктів під точними кутами та відстанями [11].

Існують різні моделі таких двигунів, як для постійного току так і для перемінного, з різною потужністю та різними призначеннями. Більш дорогі сервоприводи як правило мають набагато вищу точність та потужність, однак для даного проекту має вистачити звичайного найдешевшого.

Саме тому для регулювання куту повороту камери було використано найпростіші серво приводи SG90 (рисунок 3.3)

Рисунок 3.3 – Servo SG90

Дані сервоприводи мають невелику потужність та доволі низьку точність, але приваблюють своєю ціною, в цілому їх рекомендується використовувати для простих механізмів, які мають повертати на 90 або 180 градусів, але, як показала практика, також непогано підходять і для більш точного повороту, де потрібно повернути механізм на кут не менше 1 градусу за раз.

3.3 Потенціометри

Потенціометр – це змінний резистор, у якому контакт проходить від одного кінця резистивного елемента до іншого, у результаті чого опір пропорційний положенню контакту [12].

В даному проекті потенціометри були використані у якості джойстика для керування поворотом камери та звичайного потенціометра для зміни розміру області фіксації

Для наведення камери на ціль та регулювання розміру цілі було використано простий джойстик та потенціометер (рисунок 3.4)

Рисунок 3.4 – Аналоговий джойстик з двох потенціометрів та кнопки

До Raspberry Pi Pico даний джойстик підключається на два аналогові входи через живлення від 3.3В.

3.4 Підключення схеми

Все вище вказане було підключено до плати мікроконтроллеру через аналогові входи та PWM виходи, що можуть генерувати частотний сигнал для регуляції куту повороту сервоприводів. Система зібрана на макетній платі для тестування та налагодження роботи (рисунок 3.5).

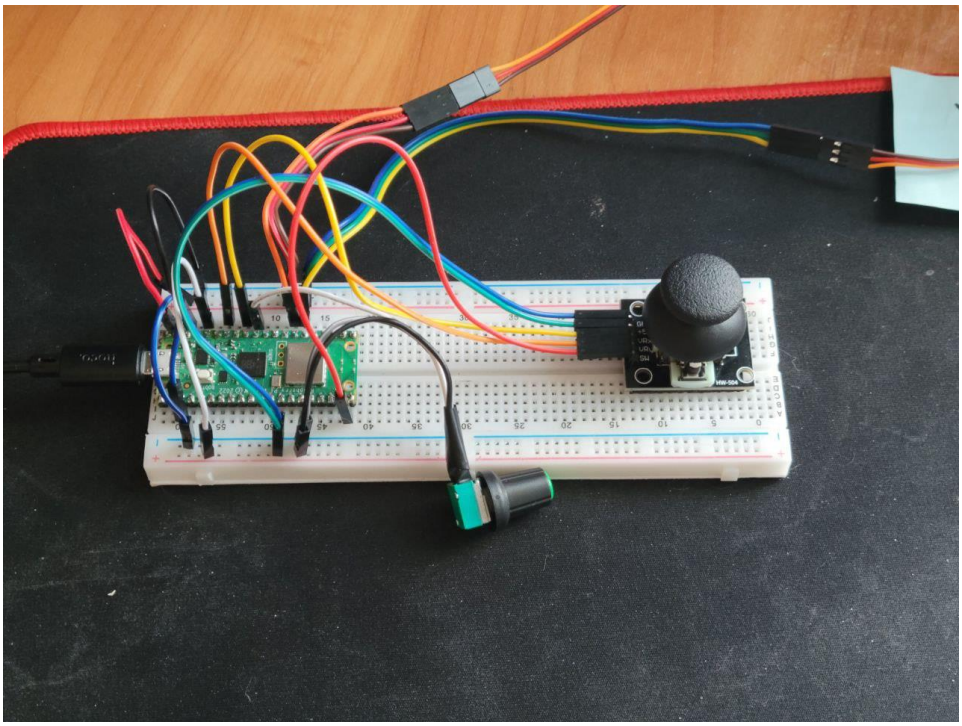

Рисунок 3.5 – Апаратна частина зібрана на макетній платі для тестування

Два потенціометра з джойстика підключаються до аналогових входів на контролері, один потенціометр для регулювання розміру зони підключається окремо на інший аналоговий вхід. Два сервопривода підключаються через звичайні піни контролера, які програмно будуть налаштовані на генерування частотного сигналу для контролю кутом повороту. Не рекомендується підключати до лінії живлення мікроконтролеру навантаження більше ніж 300мА, тому для більш потужних та точних сервоприводів потрібна буде зовнішня схема живлення.

3.5 Механізм повороту

Був розроблений та побудований механізм повороту камери по двох осях за допомогою двох сервоприводів. Механізм збирався для попереднього тестування, тому в якості каркаса використані легкодоступні міцні та легкі матеріали (рисунок 3.6).

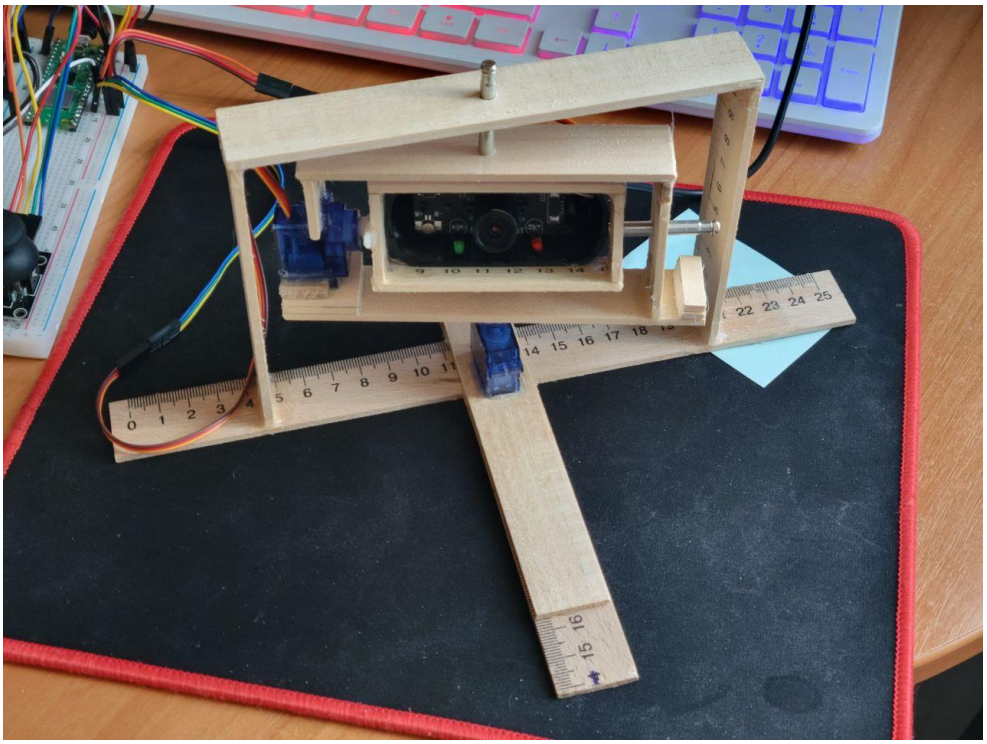

Рисунок 3.6 – Простий механізм повороту камери для тестування

Даний механізм проявив себе досить добре, тому що матеріал достатньо легкий щоб не створювати надто великої інерції при повороті, а тому поворот досить різкий та чіткий. Також хорошій роботі механізму сприяло добре налагоджений центр маси.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі будуть описані особливості програмної реалізації всієї системи, як основної програми, так і прошивки для контролеру що керує сервоприводами та синхронної роботи обох програм.

4.1 Архітектура системи

Архітектура системи складається з двох програм: основна програма для трекінгу та прошивка мікроконтролеру що керує сервоприводами, які працюють одночасно, мають два режими роботи і синхронно перемикаються, також обмінюються даними між собою по USB Serial порту. Далі будуть описані особливості роботи кожної з програм.

4.2 Програма трекер

Програма трекер – це основна програма, яка запущена на комп’ютері з потужним процесором та виконує основні обчислення щодо відслідковування траєкторії руху об’єкту. В даній програмі використовується мова програмування Python та модуль OpenCV, який був взятий за основу для реалізації слідкування за об’єктом.

При першому запуску програми запуститься цикл очікування підключення до мікроконтролеру, до того часу як програма не знайде підключений контролер та не отримає від нього правильне тестове повідомлення, для того щоб впевнитись що все працює в правильному порядку, програма працювати не буде. Далі будуть запропоновані всі доступні методи трекінгу на вибір, далі створюється клас трекера обраного методу і отримується доступ до відеокамери (рисунок 4.1).

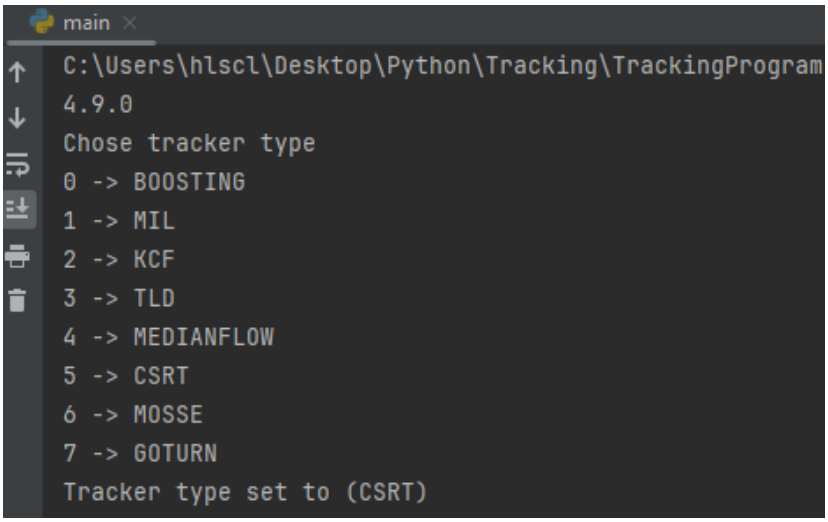

Рисунок 4.1 – Всі доступні трекери на вибір

Після створення трекеру та отримання першого зображення з камери запускається основний цикл трекінгу, в якому програма отримує новий кадр та відправляє його на обробку до функції обраного методу бібліотеки OpenCV, однак при першому запуску потрібно спочатку обрати ціль для відслідковування. Тому програма входить в режим очікування команди від мікроконтролера, а керування камерою, як вже було сказано, здійснюється за допомогою джойстика, в такому режимі програма розпізнає сигнали на зміну розміру області фіксації та очікує команду для початку відслідковування.

Після отримання команди фіксації цілі, відбувається перемикання в наступний режим, у якому буде запущений цикл трекінгу. В даному режимі після оновлення позиції об’єкту на зображенні, його позиція передається в іншу функцію, яка вираховує наскільки далеко об’єкт знаходиться від центру зображення та вираховує на який кут потрібно повернути обидва сервоприводи для того щоб приблизити об’єкт до центру зображення. Швидкість повороту сервоприводів лінійно залежить від відстані від центру зображення до об’єкту, чим далі об’єкт від центру, тим швидше відбувається поворот. Таким чином досягається максимальна точність плавність повороту камери (рисунок 4.2).

```
def convert_map_float(x, in_min, in_max, out_min, out_max):
    if x < in_min:
        return out_min
    if x > in_max:
        return out_max
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
```

Рисунок 4.2 – Функція для розрахунку швидкості повороту камери в залежності від відстані від об’єкту до центру зображення

Після того як буде вирахуваний необхідний кут повороту, дані будуть відправлені на мікроконтролер у вигляді «X:Y», де X та Y – числа з плаваючою точкою типу float, що означає кут на який потрібно повернути сервоприводи по осі по обом осям для того щоб повернути камеру на центр об’єкту.

Даний цикл автоматичного контролю поворотом камери буде виконуватись доки мікроконтролер не подасть зворотню команду для зупинки даної функції і перехід в функцію очікування.

4.2 Прошивка для керування сервоприводами за допомогою мікроконтролеру

В даній системі мікроконтролер Raspberry Pi Pico виконує функцію прослойки між основною програмою що запущена на комп’ютері та сервоприводами, однак, крім цього також він подає і зворотні команди, для зміни розміру області виділення та перехід до режиму трекінгу, а також зупинку режиму.

При запуску контролер очікує на тестовий сигнал від комп’ютеру для того щоб впевнились що команди передаються правильно, після цього він переходить до режиму ручного керування в якому поворот камерою здійснюється за допомогою джойстику, а розмір області регулюється потенціометром. По натисканню на вбудовану кнопку джойстика, мікроконтролер відправляє до

комп’ютера сигнал про перехід в режим автоматичного керування та очікує на подальші інструкції від комп’ютеру, а сам комп’ютер по отриманню даної команди переходить до режиму відслідковування об’єкту та режиму автоматичного керування камерою.

Якщо натиснути на кнопку ще раз, то контролер знову перейде в режим ручного повороту, а на комп’ютер відправиться команда для зупинки режиму слідкування.

Для контролю швидкості повороту камери в ручному режимі, в залежності від сили нахилу джойстика була використана математична функція з рисунка 4.2.

Таким чином можна використовувати систему повторно, змінювати цілі на нові і перемикати режими без потреби перезавантаження, а також керувати камерою в ручному режимі і наводити її на ціль без потреби взаємодії з самою камерою безпосередньо, що є досить зручно.

ВИСНОВКИ

У результаті проведеної роботи розроблено систему для утримання об’єкту на зображенні в заданих координатах.

Здійснено огляд та тестування різних видів трекерів, які доступні в бібліотеці OpenCV Python та оптимізація роботи програми для задовільного результату відслідковування.

Для реалізації використовувались мови Python, MicroPython, контролер Raspberry Pi Pico, потенціометри, кнопки та сервоприводи

Тестування розробленої системи показало задовільний результат. Система може відслідковувати як швидкі рухи об’єкту так і повільні, зміну його кольору та форми, змінювати ціль на іншу та задавати її початкові розміри.

Створена система може бути в подальшому вдосконалена і бути базою для впровадження в сфери де потрібні системи автоматизованого слідкування за об’єктом, такі як охоронні, оборонні підприємства, заводи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Guido van Rossum, Python Reference Manual, release 2.4.4, April 15, 2001.

2. MicroPython – Python for microcontrollers. URL: https://micropython.org (дата звернення: 26.05.2024).

3. PyCharm як найкраща IDE для розробки ПЗ на Python. URL: https://foxminded.ua/pycharm–tse/ (дата звернення: 26.05.2024).

4. Вивчення Python OpenCV. URL: https://itproger.com/ua/course/opencv (дата звернення: 26.05.2024).

5. Pyserial Documentation . URL: https://pypi.org/project/pyserial/ (дата звернення: 26.05.2024).

6. Raspberry Pi Pico vs. Arduino Nano: Which Is Best for Your Project?. URL: https://www.makeuseof.com/raspberry–pi–pico–vs–arduino–nano–best–for–project/ (дата звернення: 26.05.2024).

7. A Complete Review of the OpenCV Object Tracking Algorithms. URL: https://broutonlab.com/blog/opencv–object–tracking/ (дата звернення: 26.05.2024).

8. Microsoft – gdiplus types of bitmaps. URL: https://learn.microsoft.com/ru–ru/windows/win32/gdiplus/–gdiplus–types–of–bitmaps–about (дата звернення: 26.05.2024).

9. What is OpenCV. URL: https://blog.roboflow.com/what–is–opencv/ (дата звернення: 26.05.2024).

10. What is Arduino. URL: https://hackr.io/blog/what–is–arduino (дата звернення: 26.05.2024).

11. Fuji Electric Product Column The Purpose of Servo Motors URL: https://www.fujielectric.com/about/column/detail/servo_01.html (дата звернення: 26.05.2024).

12. Potentiometer What is a Potentiometer URL: https://www.analog.com/en/resources/glossary/potentiometer.html (дата звернення: 26.05.2024).

13. ЩО ТАКЕ МІКРОПРОЦЕСОР, МІКРОКОНТРОЛЕР URL: https://elprivod.nmu.org.ua/ua/interesting/what_is_mp_mc_plc.php (дата звернення: 26.05.2024).

ДОДАТОК А

ТЕКСТ ПРОГРАМНОГО МОДУЛЯ

«РЕАЛІЗАЦІЯ ОСНОВНОЇ ПРОГРАМИ ТРЕКІНГУ»

УКР.НТУУ “КПІ ім. Ігоря Сікорського”_ІАТЕ_ЦТЕ_ТР–02

Аркушів 4

Київ - 2024

```
#connecting with microcontroller
ser = serial_connect(port_name="COM9", timeout = 0.001)

#testing connection
test_message = "`"
serial_write(data=test_message, serial=ser)
test_request = serial_read(serial=ser)
if test_request != "hello":
    print("test connection error")
    exit(0)
else:
    print("connection success")

#set type index here to fast boot, set 8 to chose
tracker_index = 5

# Set up tracker
tracker, tracker_type, type_index = chose_tracker(index=tracker_index)

# Read video
cap = cv2.VideoCapture(2)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640) #1280
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 360) #720

# Exit if video not opened.
if not cap.isOpened():
    print("Could not open video")
    sys.exit()

# Read first frame.
```

```
ret, frame = cap.read()
if not ret:
    print(‘Cannot read video file’)
    sys.exit()

ret, frame = cap.read()
#frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
frame_height, frame_width, _ = frame.shape
frame_center = {‘x’: frame_width // 2, ‘y’: frame_height // 2}
print(f"width: {frame_width}, height: {frame_height}")

target_bool = False
frame_center = (frame_width // 2, frame_height // 2)
chose_box_radius = 50

#Draw start square in center
track_box = draw_square_in_center(frame, frame_center, chose_box_radius)
print(track_box)

turn_speed = 5
fps = 1

#Main cycle
while True:

    # Start timer
    timer = cv2.getTickCount()

    # Read a new frame
    ret, frame = cap.read()
```

```
    if not ret:
        break
    #frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)



    if target_bool:
        track_box = update_tracker(tracker, frame)
        #print(track_box)




        #print(f"turn_x: {turn_x}, turn_y: {turn_y}")
        if track_box != 1:
            turn_x, turn_y = auto_control(track_box, frame_width, frame_height,
frame_center, turn_speed=turn_speed)
            serial_write(data=f"{turn_x}:{turn_y}", serial=ser)
        data = serial_read(serial=ser)
        #print(f"get: {data}")
        if data == "fix":
            tracker, tracker_type, type_index = chose_tracker(index=type_index)
            target_bool = False



    else:
        data = serial_read(serial=ser)
        if data[0:3] == "pot":
            try:
                chose_box_radius = int(data.split(‘:’)[1])
```

```
            print(f"box radius: {chose_box_radius}")
        except Exception as e:
            print(e)

    if data == "fix":
        print(track_box)
        tracker.init(frame, track_box)
        target_bool = True
        print("target fixed")
    track_box = draw_square_in_center(frame, frame_center, chose_box_radius)
```