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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Автоматизована система динамічного визначення змін у стані об'єкта. Модуль аналізу та оптимізації.»

Виконав (-ла):
студент (-ка) IV курсу, групи КП-51
Іващенко Михайло Вікторович _____

Керівник:
Ст. викладач кафедри ПЗКС, к.т.н.,
Люшенко Л.А. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.ф.-м.н.,
Щубенкова І.А. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018 р.

ЗАВДАННЯ
на дипломний проект студенту
Іващенку Михайлу Вікторовичу

1. Тема проекту «Автоматизована система динамічного визначення змін у стані об'єкта. Модуль аналізу та оптимізації», керівник проекту Люшенко Леся Анатоліївна, к.т.н., старший викладач, затверджені наказом по університету від «22» травня 2019 р. № 1331-С.

2. Термін подання студентом проекту «14» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих рішень поставленої задачі;
   - обґрунтування вибору засобів реалізації;
   - розроблення алгоритмів та модулів;
   - аналіз розробленого ядра системи динамічного позиціонування влучень у мішень.

5. Перелік обов'язкового графічного матеріалу:
   - схема роботи алгоритму (креслення);
   - діаграма класів (креслення);
   - результати розпізнавання (плакат);
   - результати процесу навчання нейронної мережі (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	14.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Розроблення структури ядра	15.12.2018	
4.	Розгортання та тестування бібліотек, та засобів, необхідних для розробки моделі	01.02.2019	
5.	Розгортання та тестування модулю, що відповідає частині моделі «Предиктор»	20.02.2019	
6	Розробка та тестування модулю, що відповідає частині моделі «Коректор»	15.03.2019	
7.	Комплексне тестування розроблюваного ядра	14.04.2019	
8.	Підготовка матеріалів першого розділу дипломного проекту	01.05.2019	
9.	Підготовка матеріалів другого розділу дипломного проекту	06.05.2019	
10.	Підготовка матеріалів третього розділу дипломного проекту	11.05.2019	
11.	Підготовка матеріалів четвертого розділу дипломного проекту	16.05.2019	
12.	Підготовка графічної частини дипломного проекту	21.05.2019	
13.	Оформлення документації дипломного проекту	22.05.2019	

Студент М.В. Іващенко

Керівник проекту Л.А. Люшенко

АНОТАЦІЯ

Дана робота присвячена розробці системи динамічного позиціонування влучень. В якості об’єкту дослідження обрано мішень, в якості шуканих змін – влучення, отримані в результаті стрільби.

У роботі виконано порівняльний аналіз існуючих систем динамічного позиціонування різних класів, які використовуються як рішення в різних областях, розглянуто способи їх реалізації, а також існуючі недоліки. Система надає можливість виконання позиціонування змін у стані об’єкту динамічно корегуючи при цьому попередні результати в залежності від змін, що відбуваються в режимі реального часу. Збір та обробка даних здійснюється засобами пристроїв, що підключаються до мережі wi-fi та здійснюють комунікацію між собою. Ядро системи складається з двох модулів: модуль контролю (здійснює запуск системи, отримує дані з відеокамери, здійснює моніторинг сигналів на апаратному рівні, надає дані для подальшого аналізу), модуль аналізу (включає в себе нейронну мережу, яка виконує розпізнавання влучень; візуалізатор, який відображає ідентифіковані зміни всередині мішені; коннектор із базою даних, який відправляє результати в базу даних з метою виконання модифікації веб-інтерфейсу).

У даному дипломному проекті розроблено: архітектуру ядра системи, налаштовано апаратні пристрої під виконання вказаної задачі, розгорнуто алгоритми роботи з нейронною мережею (включають в себе як ті, що використовуються на стадії навчання, так і ті, що дозволяють здійснювати розпізнавання), засоби обробки результатів розпізнавання, взаємодія із базою даних, взаємодія із іншими пристроями, які присутні в системі (сервер баз даних).

ABSTRACT

This work is devoted to the development of a system of hits dynamic positioning. As a research object, the target is selected, as the required changes – hits that are obtained as a result of shooting.

In this work the comparative analysis of existing dynamic positioning systems with refer to different classes was carried out, which are currently used as solutions in various fields, the ways of their implementation are considered, as well as existing disadvantages. The system provides the ability to perform positioning changes in the state of the object dynamically adjusting the previous results, depending on the changes occurring in real time. Devices that are connected to the wi-fi network communicate with each other and perform data collection and processing. The kernel of the system consists of two modules: the control module (launches the system, receives data from the camera, monitors the signals at the hardware level, provides data for further analysis), the analysis module (includes a neural network that performs recognition of hits; a visualizer that displays identifiable changes within the target; a database connector that sends the results to the database in order to execute the modification of the web interface).

This diploma project includes: the architecture of the kernel of the system, configured hardware devices for the execution of the specified task, deployed algorithms for working with the neural network (including both those used in the training stage and those that allow the recognition), the means of processing the results recognition, database interaction, interaction with other devices that are present in the system (database server).

АННОТАЦИЯ

Данная работа посвящена разработке системы динамического позиционирования попаданий. В качестве объекта исследования выбрана мишень, в качестве искомых изменений – попадания, полученные в результате стрельбы.

В работе выполнен сравнительный анализ существующих систем динамического позиционирования нескольких классов, используемых в качестве решения в различных областях, рассмотрены способы их реализации, а также существующие недостатки. Система предоставляет возможность выполнения позиционирования изменений в состоянии объекта, динамически корректируя при этом предыдущие результаты в зависимости от изменений, происходящих в режиме реального времени. Сбор и обработка данных осуществляется средствами подключаемых к сети wi-fi устройств, которые выполняют коммуникацию между собой. Ядро системы состоит из двух модулей: модуль контроля (осуществляет запуск системы, получает данные с видеокамеры, выполняет мониторинг сигналов на аппаратном уровне, предоставляет данные для дальнейшего анализа), модуль анализа (включает в себя нейронную сеть, которая выполняет распознавание попаданий; визуализатор, который отражает идентифицированы изменения внутри мишени коннектор с базой данных, который отправляет результаты в базу данных с целью выполнения модификации веб-интерфейса).

В данном дипломном проекте разработаны: архитектура ядра системы, настроено аппаратные устройства под выполнение указанной задачи, развернуты алгоритмы работы с нейронной сетью (включают в себя как те, что используются на стадии обучения, так и те, что позволяют осуществлять распознавание), средства обработки результатов распознавания, взаимодействие с базой данных, взаимодействие с другими устройствами, которые присутствуют в системе (сервер баз данных).

ДП.045200-01-90. Автоматизована система динамічного визначення змін у стані об'єкта . Модуль аналізу та оптимізації. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045200-02-91	Автоматизована система динамічного	5	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Технічне завдання		
ДП.045200-03-81	Автоматизована система динамічного	62	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Пояснювальна записка		
ДП.045200-04-51	Автоматизована система динамічного	4	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Програма та методика тестування		
ДП.045200-05-34	Автоматизована система динамічного	8	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Керівництво користувача		
ДП.045200-06-99	Автоматизована система динамічного	1	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Схема роботи алгоритму		
ДП.045200-07-99	Автоматизована система динамічного	1	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Діаграма класів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045200-08-99	Автоматизована система динамічного	1	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Порівняння моделей SSD та YOLO		
ДП.045200-08-98	Автоматизована система динамічного	1	
	визначення змін у стані об'єкта.		
	Модуль аналізу та оптимізації.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

АВТОМАТИЗОВАНА СИСТЕМА ДИНАМІЧНОГО ВИЗНАЧЕННЯ ЗМІН У СТАНІ ОБ'ЄКТА. МОДУЛЬ АНАЛІЗУ ТА ОПТИМІЗАЦІЇ

Технічне завдання

ДП.045200-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.В. Іващенко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Автоматизована система динамічного визначення змін у стані об'єкта . Модуль аналізу та оптимізації.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості програмно-апаратного комплексу з метою зменшення витрат ресурсів та часу на збір даних з мішені та їх подальший аналіз під час стрілецьких тренувань.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Ядро системи має відповідати наступним вимогам:

- компактність апаратної системи;
- робота в режимі реального часу;
- дистанційне отримання даних з мішені;
- ідентифікація та позиціонування влучень;
- взаємодія із сервером баз даних та веб-сервером;
- розроблення в рамках ядра основ технології.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення:
   - «Автоматизована система динамічного визначення змін у стані об'єкта . Модуль аналізу та оптимізації. Діаграма варіантів використання»;
   - «Автоматизована система динамічного визначення змін у стані об'єкта . Модуль аналізу та оптимізації. Діаграма класів».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проекту.....14.11.2018
Розроблення та узгодження технічного завдання.....28.11.2018
Розроблення структури ядра.....15.12.2018
Розгортання та тестування бібліотек, та засобів.....01.02.2019
Розгортання та тестування модулю «Предиктор».....20.02.2019
Розробка та тестування модулю «Коректор».....15.03.2019
Комплексне тестування розроблюваного ядра.....14.04.2019
Оформлення документації дипломного проекту.....23.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

АВТОМАТИЗОВАНА СИСТЕМА ДИНАМІЧНОГО ВИЗНАЧЕННЯ ЗМІН У СТАНІ ОБ'ЄКТА. МОДУЛЬ АНАЛІЗУ ТА ОПТИМІЗАЦІЇ

Пояснювальна записка

ДП.045200-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.В. Іващенко

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

CPU (від англ. Central Processing Unit) – центральний процесор.

DSPU (від англ. Digital Signal Processing Unit) – процесор цифрових сигналів.

FPGA (від англ. Field-Programmable Gate Array) – програмована користувачем вентильна матриця (ПКВМ).

GPS (від англ. Global Positioning System) – глобальна система моніторингу та позиціонування.

GPU (від англ. Graphical Processing Unit) – графічний процесор.

Laser Technology Incorporation (США) – корпорація, основною діяльністю якої є розробка продуктів на основі імпульсно-лазерної технології.

Lap GmbH (Німеччина) – розробник систем динамічного позиціонування об’єктів протягом виробництва їх частин з використанням лазерних технологій.

LBL (від англ. Long baseline acoustic positioning system) – акустична система довгої базової лінії.

Pascal VOC Dataset – набір, що створюється на основі даних, які отримали маркерування у вигляді відповідного .xml-файлу, в якому вказано: модель, зону, в якій знаходиться об’єкт, який необхідно ідентифікувати та локалізувати, флагу, що визначає, чи є даний об’єкт важким для розпізнання.

Protobuf – протокольні буфери, що є нейтральними як до мови програмування, так і до платформи, і представляють собою розширюваний механізм для серіалізації структурованих даних.

SGD (від англ. Stochastic Gradient Descent) – метод стохастичного градієнту.

SVM (від англ. Support Vector Machines) – опорно-векторні машини.

V4l2loopback-модуль – модуль, що дозволяє створювати віртуальні відеопристрої, які будуть зчитувати дані з цих пристроїв так, якби вони були звичайними відеопристроями. Відеоряд при цьому має або генеруватись іншою програмою, або бути згенерованим заздалегідь.

VSPU (від англ. Vision-Specific Processing Unit) – зоровий процесор.

Zygo Corporation (США) – світовий постачальником оптичних приладів метрології, оптичних компонентів високої точності, а також послуг з проектування та виробництва електрооптичних систем.

Акустична система довгої базової лінії – акустична система позиціонування, яка відповідає одному з трьох широких класів підводних акустичних систем позиціонування, які використовуються для відстеження підводних апаратів і водолазів.

Багатошаровий перцептрон Румельхарта – окремий випадок перцептрона Розенблатта, в рамках якого алгоритм зворотного поширення помилки навчає кожний з шарів.

Виділення ознак – процес зниження розмірності, в рамках якого вихідний набір сирих даних згортається до більш керованих груп (ознак), що використовується у подальшій обробці.

Геббова теорія (Геббове навчання) – теорія в нейронауці, яка пояснює роботу нейронів мозку під час процесу навчання, описуючи механізм синапсичної пластичності.

Граф виводу – використовуються для підтримки прямого, зворотного, бінапрямленого на сфокусованого процесу отримання висновків.

Згорткова нейронна мережа – клас нейронних мереж прямого поширення, який застосовується при аналізу тексту та зображень.

Кластерний аналіз – галузь машинного навчання, що дозволяє здійснити групування даних, які не були промаркеровані, класифіковані або категоризовані. Замість того, щоб надавати відповідь на вхідні дані у вигляді класифікації, кластерний аналіз ідентифікує загальні ознаки даних і реагує

на основі наявності або відсутності тих чи інших спільних рис у кожному новому фрагменті даних.

Логічний елемент – пристрій, що використовується для обробки інформації, яка подана в цифровій формі (послідовності сигналів високого – «1» і низького – «0» рівнів у двійковій лозіці, послідовність «0», «1» та «2» в трійковій лозіці тощо).

Матричний помножувач – апаратний пристрій, що здійснює перемноження матриць за допомогою використання схеми логічних елементів.

Метод зворотнього поширення помилки – метод навчання багатошарового перцептрону, метою якого є мінімізація помилки його роботи шляхом поширення сигналів помилки від виходів мережі до її входів.

Метод стохастичного градієнта – ітеративний метод оптимізації градієнтного спуску за допомогою стохастичного наближення.

Міжнародна морська організація – міжнародна міжурядова установа в рамках ООН, діяльність якої спрямована на регулювання дій, які стосуються міжнародного торгівельного судноплавства, а також забезпечення безпеки на морі.

Область інтересу зображення – область, що є цільовою в рамках дослідження зображення на присутність об’єкта.

Опорно-векторні машини – алгоритми навчання, що реалізують в собі метод аналізу даних з метою отримання класифікації та регресійною аналізу при використанні моделей з керованим навчанням.

Пайплайн-конфігурація – конфігураційний файл, що зберігає деталізовану інформацію про структуру моделі та її параметри.

Системи динамічного позиціонування (англ. Dynamic Position Systems) – комп’ютерні системи, призначення яких полягає у виконанні стеження за певним об’єктом.

Функція втрат – функція, яка розраховується на основі нев’язки між значеннями нейронів результуючого шару, отриманих на основі вхідних даних та значеннями нейронів результуючого шару, яких мережа має досягнути внаслідок навчання.

ВСТУП

На даний момент у нашому житті існує ряд сфер, основною задачею яких є здійснення моніторингу тих чи інших параметрів об’єкту із подальшим проведенням аналізу отриманих результатів [1, 2, 3]. В даному випадку результат представляє собою зміни, які відбуваються в стані об’єкту і є викликаними впливом, який здійснюють зовнішні чинники да даний об’єкт. Прикладами таких сфер можна вважати медицину, астрономію, управління положенням об’єктів великих розмірів, системи моніторингу (охоронні системи, системи розпізнавання обличь) тощо.

Не дивлячись на широкий спектр, кожна з них вирішує задачу здійснення динамічного позиціонування змін в об’єкті в тому чи іншому вигляді. Це відбувається за рахунок використання автоматизованих систем. В їх основу можуть бути покладені різні моделі та принципи розробки, а також існує декілька підходів до їх класифікації. Тим не менш, існує ряд недоліків, які у різних поєднаннях притаманні тій чи іншій системі позиціонування. Серед них: комплексність підходу до апаратної реалізації, покладена в основу систему модель, використовувані технології, низький рівень автоматизації тощо.

Дані недоліки особливо гостро виявляються в рамках автоматизованих систем, що використовуються під час проведення спортивних або воєнних стрілецьких тренувань та змагань. Відсутність апаратної компактності (лазерні системи), гнучкості використання (SCATT, MantisX) або автоматизації як такої (використання камер для моніторингу мішені), – створюють ряд проблем, які сповільнюють та ускладнюють як процес збору даних з мішені, так і їх подальший аналіз.

В рамках даного дипломного проекту здійснюється моделювання та розроблення ядра системи динамічного позиціонування влучень у мішень із метою врахування найбільших недоліків існуючих систем та їх виправлення

або надання можливостей виправлення в рамках подальшого вдосконалення. Також наведено аналіз існуючих технологій, які використовуються в рамках сучасних систем динамічного позиціонування з метою вирішення завдань в межах різних сфер (медицина, астрономія, позиціонування об’єктів, охоронні системи тощо); розглянуто плюси та мінуси тих чи інших підходів та шляхи їх застосування в рамках розробки системи динамічного позиціонування влучень у мішень.

Для реалізації ядра використовуються такі технології, як штучний інтелект на комп’ютерний зір, на основі яких побудована модель системи. Даний документ також містить у собі деталізовану інформацію про архітектуру ядра, його модульну структуру та лістинги певних методів підмодулів; наведена оцінка продуктивності системи.

Крім цього проведено як аналіз ринкової складової системи в цілому та перспектив її комерціалізації в ролі незалежного продукту, так і можливості використання технології, покладеної в основу системи для вирішення задач динамічного позиціонування змін у стані об’єкту, які є актуальними в інших сферах.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1. Постановка задачі та проблеми динамічного позиціонування

Системи динамічного позиціонування оперують широким спектром апаратних та програмних можливостей, всі з яких використовуються для рішення основної задачі – аналізу параметрів стану досліджуваного об’єкту, що включає в себе:

- розпізнавання об’єкту (включає в себе класифікацію);
- позиціонування об’єкту у навколишньому середовищі (ідентифікація);
- відстеження та моделювання траєкторії руху об’єкту;
- відстеження та позиціонування змін у стані об’єкту.

Зазначені вище системи мають ряд обмежень:

- системи з несуттєвими розмірами об’єктів відповідно до розмірів самої системи;
- системи з обмеженням по масштабуванню та релевантності закладених математичних моделей;
- системи з обмеженнями по апаратним засобам реалізації.

Розглянемо більш детально кожне з обмежень.

Обмеження систем динамічного позиціонування з несуттєвими розмірами як об’єкту, так і змін, що відбуваються з цим об’єктом. Більшість сучасних систем орієнтується на позиціонування об’єктів та їх змін, розміри яких є масштабними відносно загального розміру «сцени», яка аналізується. Через це існуючі програмні не можуть використовуватися для задач позиціонування об’єктів (або зміни в даних об’єктах), які мають несуттєвий розмір.

Обмеження систем динамічного позиціонування відповідно до неадаптивності алгоритмів, закладених у програмних рішеннях. Дані алгоритми не є масштабованими, що прив’язує систему до конкретної

специфіки реалізації та робить задачу розширення системи складною та неоптимальною.

Обмеження систем динамічного позиціонування в апаратній частині реалізації. Апаратні пристрої в рамках систем використовуються для вирішення невідповідних для їх призначення задач. В результаті система втрачає ефективність та гнучкість.

В даному дипломному проекті запропоновано розроблення системи для вирішення задачі позиціонування влучень у мішень під час стрілецького тренування.

Задачею даної системи є аналіз мішені, в яку здійснюються постріл (при використанні камери для отримання каналу даних, які аналізуються) в режимі реального часу, конкретно: ідентифікація та позиціонування мішені на загальній сцені, ідентифікація та позиціонування влучень в мішень (якщо такі присутні). При цьому система має реагувати на зміни сцени, якщо такі відбулись (зміни світла, орієнтації мішені, орієнтації камери тощо), набуваючи ознак динамічного аналізу. Також під час роботи відбувається взаємодія із користувачем, в результаті якої оброблені дані потрапляють на пристрій за допомогою якого здійснюється підключення до системи. Остання має комбінувати у собі програмні та апаратні технології для досягнення поставленої мети.

1.2. Існуючі системи динамічного позиціонування

1.2.1. Класифікації існуючих систем динамічного позиціонування

Основною сферо використання систем позиціонування на даний момент є системи навігації, а саме: повітряної, морської. В результаті, першими, хто надав певну класифікацію систем динамічного позиціонування була Міжнародна морська організація.

Виділяють наступні класи систем динамічного позиціонування для виконання позиціонування судна [4]:

1. Клас 0 – система, позиціонування об’єкту дослідження якої відбувається вручну, а аналіз руху – автоматично за умов максимального впливу зовнішніх чинників.
2. Клас 1 – система, позиціонування об’єкту дослідження якої, а також аналіз його руху можуть відбуватись як вручну, так і автоматично за умов максимального впливу зовнішніх чинників.
3. Клас 2 – система, позиціонування об’єкту дослідження якої, а також аналіз його руху можуть відбуватись як вручну, так і автоматично за умов максимального впливу зовнішніх чинників та при наявності несправностей при паралельному використанні двох екземплярів системи.
4. Клас 3 – система, позиціонування об’єкту дослідження якої, а також аналіз його руху можуть відбуватись як вручну, так і автоматично за умов максимального впливу зовнішніх чинників та при наявності несправностей при паралельному використанні принаймні двох екземплярів системи, для кожного з яких наявна можливість відновлення з резервної копії.

Також виділять класифікацію за типом системи позиційного контролю, яка використовується. У більшості випадків системи динамічного позиціонування використовують більше, ніж одну систему моніторингу. Рішення, що використовують менше, ніж три системи моніторингу прийнято відносити до Класів 0 або 1, інші – до Класів 2 та 3.

Три системи використовують для уникнення наступних ефектів:

- «Неможливо визначити, яка з двох систем видає неправильний результат» у випадку, коли аналіз відбувається двома системами.
- «Неможливо перевірити результати роботи системи» у випадку, коли аналіз відбувається одною системою.

1.2.2. Класифікації існуючих систем динамічного позиціонування

Нижче наведені типи найпопулярніших систем моніторингу, що використовуються у сучасних системах динамічного позиціонування:

1. Диференціальна глобальна моніторингу та позиціонування – розширення системи глобального позиціонування GPS. Псевдодіапазони, отримані приймачем, порівнюються із аналогічними псевдодіапазонами, отриманими з супутників та опорних станцій з метою забезпечення корекції результатів. Дані системи забезпечують визначення позиції досліджуваного об’єкту з точністю від 15 метрів до 10 см.
2. Гідроакустична система моніторингу та позиціонування – система для вимірювання відносного положення між передавачем і приймачем під водою. Використовується для управління судами за рахунок відстеження їх позицій, а також вимірювання взаємного розташування між фіксованим підводним передавачем або мобільним підводним транспортним засобом, враховуючи положення базових ліній. Базові лінії являють собою тривимірну лінію, проведену між двома точками, зайнятими кожною парою GPS-антен. Перетворювач посилає акустичний сигнал до пристрою, що ініціює відповідь. Оскільки швидкість звуку через воду відома, відстань відома, так як на датчику присутня велика кількість елементів, можна визначити напрямок сигналу від пристрою. Точність даних систем складає від 1 метру до 3 см.
3. Лазерні системи моніторингу та позиціонування – системи, що використовують лазер для аналізу та позиціонування об’єкту. Широко використовуються як у сфері управління судами, так і у воєнній сфері. За допомогою лазерів здійснюється аналіз одної або ряду цілей (за умови, що вони зберігають загальну структуру). Пристрій надсилає світлові імпульси, які приймаються таким

чином, щоб можливо було визначити їх діапазон. За рахунок технологічної специфіки лазеру, мають найвищу точність – до 4 мм.

4. Охоронні системи моніторингу та позиціонування – системи, що виконують спостереження за станом певної області. Більшість з них надає наступні можливості: нічний зір, відео у форматі HD, автоматичні панорамування, нахил та масштабування, дистанційне керування з мобільного пристрою. Рідше зустрічаються системи, всередині яких знаходиться програмне забезпечення, яке дозволяє ідентифікувати зміни на сцені та відстежувати рух.

Розглянемо більш детально існуючі системи динамічного позиціонування та їх призначення.

Системи управління, побудовані на основі GPS-позиціонування

Загалом у світі до 47 країн використовують системи позиціонування, побудовані на GPS-технології. GPS-вимірювання зазвичай зберігаються в пам'яті комп'ютера в GPS-приймачах і згодом передаються на комп'ютер, що працює з програмним забезпеченням GPS-обробки. Програмне забезпечення обчислює базові лінії за допомогою одночасних даних вимірювань від двох або більше приймачів GPS. Пост-оброблені вимірювання дозволяють більш точне позиціонування, оскільки більшість помилок GPS впливають на кожного приймача майже однаково, і тому ними можна знехтувати при виконанні розрахунків.

Системи управління, побудовані на основі систем гідроакустичного позиціонування

Одною з найбільших LBL-систем у світі є система Fusion 6G виробником якої є компанія Sonardyne. Система виконує вимірювання акустичних діапазонів до масиву транспондерів на морському дні. Потім діапазони передаються через обчислення найменших квадратів, щоб точно виділити позицію. Оскільки система використовує фіксований масив морського дна, точність системи залишається незмінною незалежно від

глибини води. Також система оснащена програмним забезпеченням, яке підтримує різноманітні інструментальні платформи.

Системи управління, побудовані на основі систем лазерного позиціонування

Найяскравішими прикладами даних систем є рішення, які дозволяють налаштовувати вісі координатних систем під час проектування. Даний метод є корисним при виробці деталей, використанні пристроїв, призначених для перевірки якості води та інших систем, в яких важливою є орієнтація пристрою відносно досліджуваного об'єкта. Одними з компаній, які розробляють дані системи є Laser Technology Incorporation, LAP GmbH, Zygo Corporation.

Охоронні системи моніторингу та позиціонування

Основними пристроями, що формують даний тип систем є різноманітні IP-камери (гніздові та кільцеві камери), з'єднані у єдиний механізм, який дозволяє вести спостереження за всією площею. Іноді, до камер долучають програмне забезпечення, що дозволяє слідкувати за рухом (прикладом одної з відкритих систем є Motion).

В рамках зазначеної класифікації систему, опис якої наведено в підрозділі 1.1, має бути побудована за принципами охоронних систем моніторингу та позиціонування. Тоді як останні є системами, що відносяться до Класу 1, описана система має бути реалізована за принципами, покладені у системи Класів 2 та 3, або реалізовувати алгоритм ідентифікації та динамічного позиціонування таким чином, щоб уникнути ситуації, в якій неможливо перевірити кінцевий результат аналізу даних. Крім цього, вона має містити у собі адаптивні та ресурсномісткі алгоритми обробки.

1.3. Програмні та апаратні засоби для створення автоматизованих систем динамічного позиціонування

1.3.1. Технології для розробки програмного забезпечення

Основними програмними технологіями, які використовуються для створення автоматичних систем позиціонування є:

- комп’ютерний зір;
- машинне навчання;
- штучний інтелект.

Розглянемо більш детально кожну з названих технологій.

Комп’ютерний зір є міждисциплінарною науковою сферою, яка стосується того, як комп'ютери можуть бути створені для отримання високого рівня розуміння з цифрових зображень або відео [5]. З точки зору інженерії, вона прагне автоматизувати завдання, які може робити візуальна система людини. Не зважаючи на широкий спектр використання, системи комп’ютерного зору, містять в собі ряд типових функцій:

1. Отримання зображення. Система або власноруч формує зображення в цифровому форматі на основі формалізації та узагальнення інформації, отриманої з різних датчиків, або використовує готові пристрої (наприклад, камери). Результат може бути змодельовано у вигляді 2D або 3D-зображення. Кожна точка такого зображення, що є його елементом, називається пікселем. Значення пікселя є результатом оцінки декількох величин, основною з яких є інтенсивність світла в одній або декількох спектральних полосах. Крім цього, воно може бути отримане як результат виміру глибини, відбиття або інших явищ, що відносяться до світла.
2. Передчасна обробка. Між стадіями отримання зображення та безпосереднім виконанням алгоритмів комп’ютерного зору

здійснюється перевірка характеристик даного зображення на задовільність умовам, що мають виконуватись для використання того чи іншого алгоритму обробки.

3. Виділення деталей. Відбувається виділення характерних для зображення деталей, таких як контури, кути, області інтересу тощо.
4. Детектування та сегментація. Етап обробки полягає у прийнятті рішення з приводу того, які точки є характерними по відношенню до даних, що були отримані на попередніх кроках. Після цього відбувається виділення ділянок зображення, що містять шуканий об’єкт.
5. Виконання обробки відповідно до певного рівня. Після завершення сегментації оброблені дані використовуються як вхідні параметри для певних методів обробки зображень або повноцінних алгоритмів комп’ютерного зору.

В рамках створення автоматичної системи, необхідної для виконання динамічного позиціонування недостатньо лише засобів комп’ютерного зору. Необхідні корекція та підтвердження отриманих результатів аналізу.

Машинне навчання – це вивчення та побудова алгоритмів і статистичних моделей, які використовують комп'ютерні системи для ефективного виконання конкретних завдань без використання явних інструкцій, використовуючи замість них заздалегідь побудовані моделі та методи. Дана технологія розглядається як підтип штучного інтелекту. Алгоритми машинного навчання будують математичну модель вибіркових даних, відомі як "дані навчання", для того, щоб отримувати прогнози або висновки, не будучи явно запрограмованими для виконання відповідного завдання.

Існує два види навчання:

1. Навчання з вчителем – вид машинного навчання функції, яка будує відповідність між вихідними даними та результуючими даними, яка

представляє себе в парах введення-виведення [6]. Він синтезує функцію на основі маркерованих даних тренувань, що складаються з набору навчальних прикладів [7]. У контрольованому навчанні кожен приклад являє собою пару, що складається з вхідного об'єкта (як правило, вектора) і бажаного вихідного значення (також званий контрольним сигналом).

2. Напівавтоматичне навчання – клас задач і методів машинного навчання, які використовують невелику кількість маркерованих даних з великою кількістю немаркерованих даних даних. Принцип напівавтоматичного навчання позиціонує себе між навчанням без вчителя (маркерування даних відсутнє) та із вчителем (дані повністю промаркеровані). Багато дослідників машинного навчання виявили, що немаркеровані дані, коли вони використовуються разом з невеликою кількістю позначених даних, можуть значно поліпшити точність навчання.
3. Навчання без вчителя – термін, що використовується для Геббового навчання, і асоціюється з навчанням без вчителя, також відомого як самоорганізація і метод моделювання щільності ймовірності входів за рахунок використання кластерного аналізу [8].

Штучний інтелект. У галузі комп'ютерних наук штучний інтелект, є інтелектом, що демонструється машинами, на відміну від природного інтелекту, що проявляється у людини або. Комп'ютерні науки визначають дослідження штучного інтелекту як дослідження та формалізація проблем і завдань, подібні до дій, які виконує людина: будь-який пристрій, який сприймає своє середовище і вживає заходів, які максимізують його шанс на успішне досягнення своїх цілей. Більш конкретно, Каплан і Ганлейн визначають штучний інтелект як «здатність системи правильно інтерпретувати зовнішні дані, вчитися за такими даними і використовувати

ці знання для досягнення конкретних цілей і завдань за допомогою гнучкої адаптації» [9].

Одним з інструментів реалізації штучного інтелекту є використання нейронних мереж, які стали основним методом в рамках виконання задач розпізнавання об’єкту із високою точністю. Для організації навчання нейронних мереж існує ряд моделей розпізнавання об’єктів. Для вирішення задач, що будуть поставлені в рамках дипломного проекту, розглядались варіанти наступних моделей:

R-CNN

Дана модель складається з трьох основних модулів [10]: перший генерує набір регіонів інтересу, які є незалежними від категорії; другий представляє собою велику згорткову нейронну мережу, яка виділяє вектор ознак кінцевої довжини; третій є набором залежних від класів алгоритмів навчання з вчителем.

На відміну від попередніх версій моделі CNN [11], R-CNN здійснює обмежений вибір регіонів, з яких в подальшому буде виділятись вектори ознак. Даний підхід виділяє не більше, ніж 2000 регіонів з вихідного зображення, які називаються запропонованими регіонами (або регіонами-кандидатами). Дані регіони генеруються на основі використання вибіркового алгоритму пошуку, логіка роботи якого закладена в основи нейронної мережі (RPN), за результатами роботи якої здійснюється відбір. Сам алгоритм складається з наступних кроків [12]:

1. Створивши початкову підсегментацію, генеруємо регіони-кандидати.
2. Використовуємо жадібний алгоритм, рекурсивно комбінуючи однакові або приблизно однакові регіони в регіони, які є більшими за розміром.
3. Використовуємо згенеровані регіони для формування фінальних запропонованих регіонів.

Ці 2000 запропонованих регіонів-кандидатів згортаються у квадрат і надходять до нейронної мережі, яка повертає 4096-мірний вектор ознак як вихідний. CNN виступає в якості екстрактора ознак, її кінцевий щільний шар складається з ознак, отриманих з зображення, і тих, що подаються в SVM для класифікації присутності об'єкта в запропонованому регіоні-кандидаті. Крім цього, з метою підвищення точності прогнозування присутності об'єкта в межах запропонованого регіону, алгоритм повертає чотири значення позиції об'єкту, які несуттєво зміщені одне відносно одного (див. рис. 1).

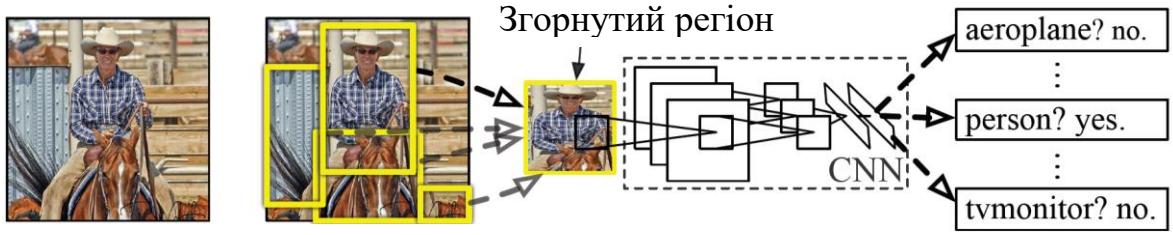


Рис. 1. Алгоритм роботи моделі R-CNN

Також існує більш оптимальна та швидка версія R-CNN, яка називається “Faster R-CNN” [13]. Її ідея полягає у виконанні етапів формування регіонів-кандидатів та розрахунку ознак нейронними мережами, які є згортковими (див. рис. 2).

SSD

Основою моделі SSD є згорткова нейронна мережа (що також представляє собою багатошаровий перцептрон Румельхарта), яка повертає набір обмежуючих боксів, що мають фіксований розмір, та відповідних ним оцінок присутності в них того чи іншого шуканого класу об’єкту. Перші шари мережі моделюються у відповідності зі стандартною архітектурою, яка використовується при розпізнаванні об’єкта на сцені, що також називаються «базовою мережею» [14].

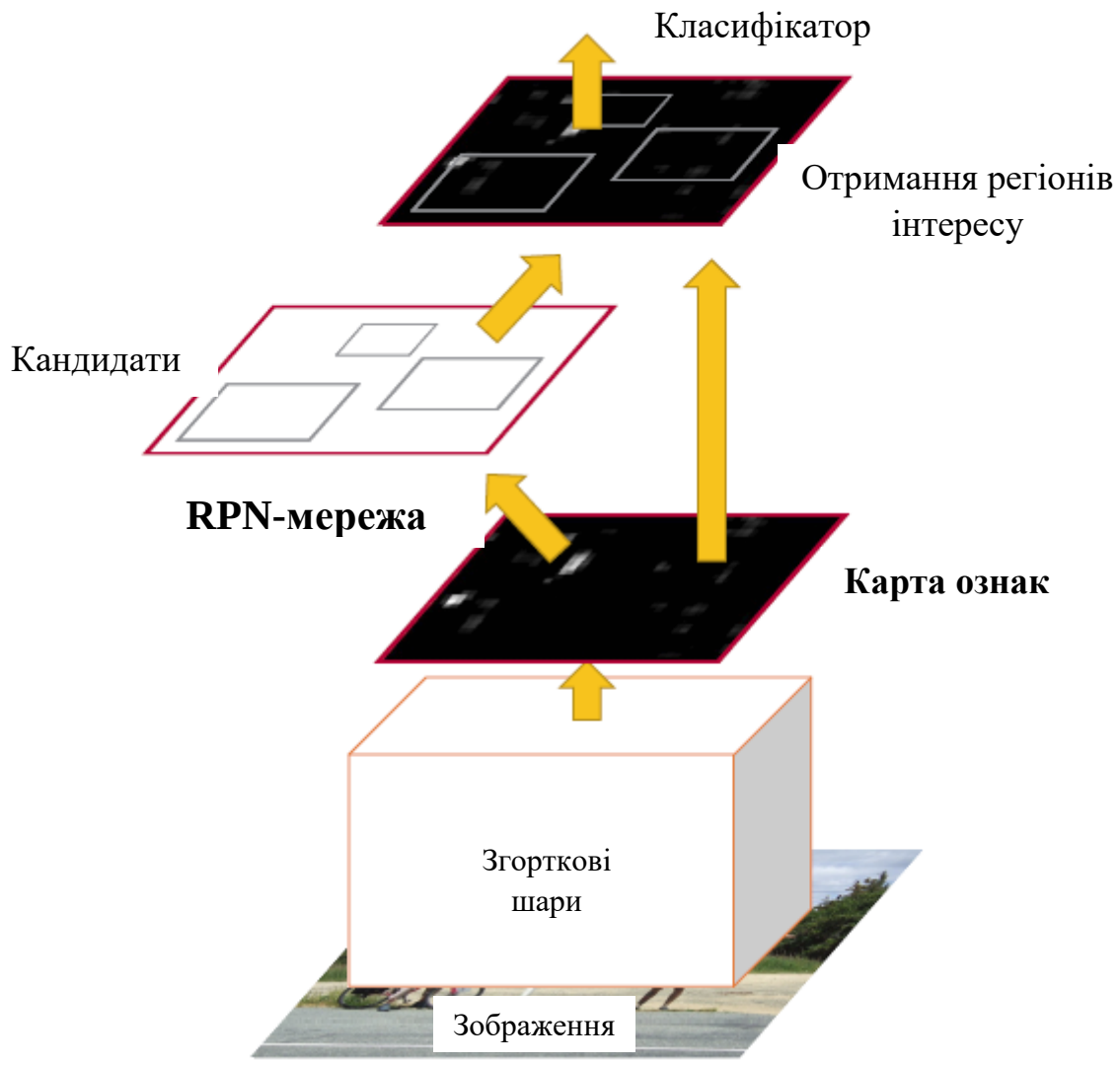


Рис. 2. Алгоритм роботи Faster R-CNN моделі

Таким чином, на відміну від R-CNN, дана модель використовує тільки одне зображення для отримання результатів класифікації (тоді як в рамках R-CNN один екземпляр необхідний для отримання регіонів-кандидатів, інший – для виконання розпізнавання), роблячи її суттєво швидшою за попередників (Додаток 1).

SSD реалізовує в собі наступний алгоритм розпізнавання:

1. Після отримання результатів виділення ознак, формується вихідний шар розміру *mxn* із кількістю каналів *p*. Згортка виконується до розміру 3x3.

2. Для кожного об’єкту отримуємо k обмежувальних боксів, для кожного з яких відбувається розрахунок c класів та 4 зміщення відносно обмежувального боксу, встановленого за замовченням.
3. Так чином, отримуємо $(c+4)kmn$ виходів.

Основною відмінністю між тренування SDD та інших моделей розпізнавання, які використовують побудову набору регіонів кандидатів, полягає в необхідності наявності базової розмітки, що відповідає класам об’єктів, пошук яких планується виконувати. Як тільки це зроблено, відбувається розрахунок функції втрат та робота методу зворотнього поширення помилки.

На даний момент системи динамічного позиціонування рідко використовують штучний інтелект або машинне навчання, тим не менш, поступово з’являються пропозиції того, яким самим чином це можна реалізувати [15, 16].

1.3.2. Апаратні технології

Автоматичні системи динамічного позиціонування, що побудовані на основі програмних засобів можуть використовувати ряд різноманітних процесорів для виконання обчислень та аналізу отриманих даних. Основними використовуваними процесорами є центральні процесори та графічні процесори. Тим не менш, крім даних процесорів існує ряд процесорів, що рідше, але використовуються у подібних системах (див. рис. 3).

Рис. 3. Основні види процесорів, що використовуються в системах комп’ютерного зору

Центральний процесор – частина комп’ютера, що виконує інтерпретацію команд і вміщую у себе наступні функції: обробка даних та виконання арифметичних і логічних операцій, виконання програмного керування роботою пристроїв, які знаходяться всередині комп’ютера. Дані процесори можуть використовуватись для виконання обчислень алгоритмами комп’ютерного зору. Тим не менш, центральні процесори не є найоптимальнішим інструментом через певні обмеження. Найвища ефективність їх роботи досягається при виконанні складних операцій, зв’язаних із прийняттям рішень, загальним контролем або пришвидшенням певних дій, пов’язаних із обробкою на піксельному рівні.

Графічний процесор – частина комп’ютера або іншого мультимедійного пристрою, що призначений для виконання графічної обробки та рендерингу. Основна відмінність від центрального процесору полягає в архітектурі, що максимально націлена на підвищення швидкості виконання обчислень при використанні текстур та складних графічних об’єктів, а також в обмеженому списку команд. Високопродуктивні графічні процесори володіють можливістю виконувати велику кількість паралельних обчислень (при використанні організацій графічних процесорів дані обчислення інакше називаються обчисленнями загального призначення на графічних процесорах), що робить їх цінними пристроями в системах комп’ютерного зору. У більшості випадків вони використовуються у зв’язках із центральними процесорами, беручи на себе частину функцій обробки та залишаючи останнім функції контролю над собою.

Програмована користувачем вентильна матриця або ПКВМ – напівпровідниковий пристрій, який може бути налаштований користувачем після придбання або виробником (розробником) після виготовлення. Дана мікросхема складається з програмованих вентилів та програмованих з’єднань між ними. Замість виконання високовартісних та довгих операцій, вона представляє їх у вигляді міні-схем, що складаються з простих логічних

операцій, і найголовніше – виконуються з такою швидкістю, яка дозволяє вважати роботу системи близькою до тої, яка була б повністю складалась з паралельно виконуваних операцій. На відміну від центральних та графічних процесорів, ПКВМ не може (і не має) обробляти багатопоточних або розподілених у часі задач, так як при використанні подібних організацій системи, задачі всередині неї будуть витрачати час на вирішення пріоритезації виконання. Організація даних пристроїв дозволяє суттєво пришвидшити операції, необхідні для виконання під час роботи систем комп’ютерного зору.

Зорові процесори – клас мікропроцесорів, що почав започатковуватись у 2016 році. Вони відрізняються від інших процесорів своєю адаптивністю при виконанні алгоритмів машинного зору (згорткові нейронні мережі, незалежне від масштабу перетворення ознак тощо). Основна увага роботи даних процесорів сконцентрована на роботі із потоками інформації, але, як і інші процесори, вони можуть бути налаштовані на обробку операцій, що включають в себе арифметику з нерухомою комою низької точності (широко використовується при обробці зображень). Основною відмінністю зорових процесорів від графічних є відсутність спеціалізованого апаратного забезпечення та оптимізованої структури, призначеної для виконання маніпуляцій із растровими зображеннями. Крім цього, в порівнянні із іншими процесорами, вони досягають більшого енергозбереження відносно своєї собівартості, реалізовуючи в собі систему ко-процесорів та прискорювачів. Основним недоліком даних процесорів є необхідність їх використання та налаштування виключно для певної конкретної задачі.

Процесор цифрових сигналів – спеціалізований мікропроцесор, що є програмованим і призначений для виконання маніпуляцій із потоками цифрових даних в режимі реального часу. Під час обробки значення сигналу може бути задіяне в арифметичних операціях або фільтрації. В результаті,

зображення може бути представлене як група точок, що визначається двовимірними координатами та в подальшому оброблене.

В рамках розглянутої класифікації, а також апаратних недоліків існуючих систем, система, реалізація якої поставлена в якості основного завдання, має комбінувати у собі різні типи процесорів для досягнення максимальних ефективності та продуктивності роботи. Відповідний клас задач має бути розгорнутий на відповідному типу процесорів, а саме: задачі класифікації – CPU, задачі графічної обробки – GPU, задачі розбору даних та штучного інтелекту – FPGA.

1.4. Обґрунтування розробки

Спільні ознаками описаних систем динамічного позиціонування є або їх апаратна залежність від певного набору пристроїв (GPS або гідроакустичні системи), або специфіки задач, вирішення яких потребує реалізації програмованого ядра.

Зазвичай, дана специфіка полягає у необхідності ідентифікації або розпізнання об’єктів, розміри яких відносно загального розміру зображення можна вважати суттєвими (прикладом є розпізнання певних об’єктів, руху тощо). При цьому, основним допоміжним фактором для системи є відносно масштабні зміни, що відбуваються на сцені. Але що буде відбуватись, якщо необхідно вести спостереження сцени, об’єкти на якій мають малі розміри відносно неї? Задачу, що витікає з даного питання можливо формулювати наступним чином: маємо об’єкт, що ідентифікує собою певну сцену, і в змісті якого через певні проміжки часу відбуваються певні зміни; необхідно не тільки ідентифікувати факт, що зміни відбулись, а й виконати позиціонування даних змін в об’єкті (на сцені).

Прикладом систем, що імітують вирішення даної задачі є охоронні системи. Тим не менш, алгоритми, що використовуються у даних системах є пристосованими виключно для ідентифікації відносно масштабних змін на

сцені, і не можуть бути використані у таких задачах, як: спостереження за рухом небесних тіл, спостереження за рухом клітин при використанні мікроскопу тощо. Несуттєві коливання світла, рух об’єкта, за яких ведеться спостереження (зміна орієнтації камери), суттєві зміни загального стану об’єкта, що можуть бути викликані навіть важкими погодними умовами, – все це створює вплив на дані системи, змушуючи їх реагувати. Відповідно, необхідним є або адаптація даних алгоритмів до ідентифікації та позиціонування мінімальних змін всередині об’єкта, або розроблення алгоритму для виконання даних задач.

1.5. Висновки до розділу

В якості основної задачі було поставлено розроблення ядра системи динамічного позиціонування. Досліджуваним об’єктом була обрана стрілецька мішень, в яку здійснюються постріли. Система має виконувати ідентифікацію та позиціонування змін в її стані, які відбуваються внаслідок влучень. Організація підсистем має забезпечувати набуття основною системою ознаки «динамічна».

Відповідно, можливо виділити три категорії вимог для ядра системи:

1. Загальні вимоги:
    - Система виконує ідентифікацію та динамічне позиціонування як досліджуваного об’єкту на сцені (мішень), так і змін в його стані (влучення).
    - Система реалізовує у собі принципи, закладені в системах Класів 2 та 3 (або подібні до них).
    - Система побудована за спрощеною моделлю у порівнянні з існуючими аналогами.
2. Програмні вимоги:
    - Система реалізовує алгоритми, що забезпечують вищу ефективність в порівнянні із існуючими аналогами.

- Система реалізовує в собі адаптивні алгоритми (під адаптивністю розуміється відсутність потреби формування алгоритмів виключно для конкретно обраної задачі).
- Система реалізовує алгоритми, в яких закладений комбінований підхід до використовуваних технологій для отримання відповідного поєднання точності та ефективності.

Окрім цього, говорячи про потенційне вдосконалення системи, варто виділити вимогу до апаратної частини, які мають бути реалізовані:

- В рамках системи реалізація задач має відповідати конкретним типам процесорів.
- Система забезпечена апаратною структурою, в рамках якої будуть комбінуватись декілька типів процесорів (за умови, що цього вимагає поставлена задача).
- Алгоритми, реалізовані в рамках програмної частини є адаптованими під типи процесорів, що використовуються.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Програмні технології та засоби для побудови автоматизованої системи

З метою підвищення рівня оптимізації роботи системи, а також її загальної ефективності, необхідно використовувати відповідну модель в рамках якого буде здійснюватись розроблення.

2.1.1. Модель «Предиктор-Коректор»

За вимогою, сформованою у пункті 1.5, система має відповідати характеристикам систем Класів 2 та 3. Для виконання даної вимоги була обрана наступна модель: «Предиктор-Коректор».

Як реалізація, так і використання даної моделі є оптимальними за рахунок того, що обидві її частини формують підсистеми, які можна віднести до Класу 3. Це означає, що кожна з них містить у собі ряд методів та підходів, спроможних не тільки отримувати результат проміжної обробки, а й оцінювати ступень його коректності, порівнюючи при цьому результати один одного. Тим не менш, основною перевагою даної моделі є загальний підхід виконання аналізу. Зібрані дані проходять не одну стадію підтвердження коректності, а дві:

«Предиктор» збирає дані, виконуючи первинні аналіз та підтвердження наявності даних для подальшого корегування. Дана стадія вміщує в собі алгоритми передчасної обробки. Її основною задачею в рамках загальної системи є підготовка отриманих даних до процесу обробки, а виконання первинного позиціонування всіх змін, що відбулись у стані об’єкту.

«Коректор» отримує оброблені дані з метою виконання підтвердження наявності шуканого об’єкту. Даний етап вміщує в себе задачі ідентифікації та класифікації, а також, у деяких випадках, моделювання на

основі статистичних показників. В рамках системи основною задачею даної стадії є виділення шуканих змін поміж позиціонованих (в даному випадку – виділення влучень).

2.1.2. Інформаційні технології для побудови автоматизованої системи

Основними технологіями, що були обрані для задоволення вказаних вимог, було вибрано:

1. Штучний інтелект – втілює собою роботу «Коректора». «Предиктор», виконуючи первинну оброку, відмічає місця всередині об'єкту, в яких відбулись зміни. Задачею «Коректора» є підтвердження, що та чи інша зміна є влученням; для реалізації даного механізму, «Коректор» реалізовано як нейронну мережу, налаштовану на пошук влучень в середині мішені.
2. Комп'ютерний зір – алгоритми комп'ютерного зору дозволяють оптимально виконувати первинну обробку та підготовку зображення. Елементи даної технології розгорнуті всередині «Предиктора». Він виділяє області, в межах яких було виконано позиціонування зміни стану. Результати надсилаються Коректору.
3. Математична обробка – використання математичних алгоритмів для оптимізації загального процесу обробки. Основною задачею для вирішення якої в системі використовується математична обробка є зменшення область інтересу, в рамках якої здійснюється дослідження даних на наявність змін.

2.1.3. Вибір мов програмування

Мови програмування були використані для реалізації як алгоритмів «Предиктора», так і «Коректора».

Для вирішення задач первинної обробки зображення, його підготовки, а також виконання початкового позиціонування змін, які реалізовані всередині «Предиктора», було обрано наступні мови програмування:

1. C++ (версія компілятора gcc – 8.2.0 або вище) – це мова програмування загального призначення, розроблена Бьярном Страуструпом як розширення мови C, або "C з класами". C ++ був розроблений в напрямку використання системного програмування, а також побудови вбудованого програмного забезпечення з обмеженими ресурсами та великих систем, з достатньою продуктивністю, ефективністю та гнучкістю використання. C++ є одною з найшвидших та оптимізованих мов програмування для виконання алгоритмів обчислювальної обробки, а також для вирішення задач класифікації або управління бізнес-логікою системи в цілому.
2. В рамках системи C++ використовується для програмування базових шарів ядра, а саме: початкова обробка даних, отриманих з вибраного джерела; взаємодія із базою даних на різних етапах роботи системи (включає в себе як елементи комунікації, так і обміну даними); бізнес-логіка роботи системи (відповідає за повний контроль над процесами, що відбуваються всередині системи); виконання ролі інтегратора при взаємодії із іншими модулями системи (C++ успішно взаємодіє з C та Python, кожна з яких розширюють загальні можливості як системи в цілому, так і її управлінням).
3. Python (версії 2.7.0 і вище) – є інтерпретованою мовою програмування високого рівня загального призначення, яка була створена Гвідо ван Россумом. Python реалізує в собі динамічну типізацію та автоматичне керування пам'яттю. Він підтримує багато парадигм програмування, включаючи об'єктно-орієнтовані,

імперативні, функціональні та процедурні. Він володіє чисельною стандартною бібліотекою. Його основні парадигми, які застосовуються як правила написання коду, описані в документі “Zen of Python” [17].

Замість того, щоб зберігати всі свої функціональні можливості вбудованими в ядрі, Python був розроблений, щоб бути максимально масштабованим та переносним. Саме ці характеристики, а також компактна структура модулів зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих додатків. Тим не менш, основною перевагою Python-у є можливість його використання при організації оболонок роботи системи (або її підсистем). Під оболонкою мається на увазі комплекс скриптів, організація яких дозволяє налаштовувати управління системою (підсистемою) та порядок виконання її модулів; Python надає інтерфейс, що дозволяє зробити роботу з даними скриптами максимально легкою за рахунок можливості взаємодії з ними через використання команд CL-інтерфейсу.

Будучи успішно інтегрованим з C/C++, Python дозволяє створювати оболонки до існуючих реалізацій. Даний підхід використовують з метою полегшення як засобів, що використовуються для управління програмою, так і програмного коду в цілому. При цьому, швидкість роботи отриманих модулів є близькою до оригінальних, але значно полегшується інструментарій управління та інтеграції в загальні системи.

Саме ця можливість Python використовується в рамках системи (як в «Предикторі», так і в «Коректорі»). В рамках Предиктора, скрипти Python-у реалізують в собі алгоритми комп’ютерного зору, відповідальні за виконання первинної графічної обробки. Даний набір скриптів запускається за допомогою одної команди, сигнатура якої складається з виклику модулю Python, системи модулів, які необхідно запустити та системи даних, які використовуються для коректної роботи або аналізу. В рамках «Коректора»

Python використовується для організації нейронної мережі, а також процесів, відповідальних за її навчання. Завдяки внутрішній конфігурації, яка закладена в даній мові програмування, процеси роботи зі штучним інтелектом є оптимізованими і не потребують знаних витрат часу та ресурсів на їх виконання.

2.2. Апаратні засоби для побудови автоматизованої системи

2.2.1. Обрані процесори

Вибір процесорів для потенційного вдосконалення системи здійснювався в рамках поставлених задач, виконання яких потребує різну специфіку апаратної структури. Розглянемо обрані процесори відносно задач, які виконується в процесі роботи одного циклу системи.

Отримання даних та їх підготовка для передачі в основну систему

Для виконання даної задачі в рамках системи використовуються процесори цифрових сигналів. Даний тип процесорів було обрано через специфіку каналу вхідних даних, роль якого виконує відеопотік. Зображення представляється групою точок, яка задіюється в операціях обробки та фільтрації як цифрові дані. При цьому за умови використання процесорів цифрових сигналів швидкість роботи алгоритмів підготовки даних буде витрачати найменшу кількість ресурсів та часу.

Виконання базової графічної обробки

Для організації графічної обробки в рамках стадії підготовки даних для подальшого позиціонування змін, було обрано графічні процесори. Спеціалізована архітектура дозволяє підвищити ефективність виконання алгоритмів графічної обробки за рахунок можливостей організації великої кількості паралельних обчислень.

Організація та робота нейронної мережі

Специфіка нейронної мережі полягає у наявності великої кількості нейронів та синапсів, які в сумі створюють модель, подібну до людського мозку. Для отримання найвищої ефективності, роботу даної моделі необхідно імплементувати на основі процесорів, що дозволили б максимально розпаралелити роботу. Саме тому для вирішення даної задачі були обрані процесори на основі ПКВМ-плат. ПКВМ-плата дозволяє побудувати з реалізованої на неї логіки паралельну схему, що складається з чисельної кількості логічних елементів. Дана схема є подібною до базових концепцій організації нейронної мережі, що дозволяє суттєво пришвидшити як процес навчання, так і безпосередній процес роботи мережі.

Виконання задач класифікації, розподілення пам'яті та прийняття рішень

Для виконання даного спектру задач було обрано центральний процесор, ресурс та можливості якого відповідають виконанню алгоритмів загального призначення. Даний тип процесорів пристосований до того, щоб витримувати суттєве навантаження протягом всього часу роботи систем, а багатоядерність подібних процесорів дозволяє організацію паралельних обчислень. Основною перевагою використання центральних процесорів з метою реалізації на їх основі задач класифікації, розподілення пам'яті та прийняття рішень є те, що дані задачі мають виконуватись протягом всього циклу роботи системи.

2.3. Висновки до розділу

Для задоволення вимог до проекту, наведених в підрозділі 1.5, було розглянуто та обрано технології реалізації.

Модель «Предиктор-Коректор»

Для організації «Предиктора» використовувалися алгоритми та принципи математичної обробки та комп'ютерного зору. Внаслідок цього

«Предиктор» виконує первинне позиціонування області, в рамках якої здійснюється подальше розпізнавання. Для реалізації даного блоку використовувались засоби мов C, C++ (версія компілятора gcc 8.2.0) та Python версій 3.5 та 3.6.

«Коректор» виконує розпізнавання та позиціонування влучень, використовуючи штучний інтелект (а саме – нейронну мережу), навчену в рамках використання моделі моделі SSD, що здійснює навчання нейронної мережі із вчителем. Організація та навчання нейронної мережі здійснювалось за рахунок використання бібліотек та API для роботи зі штучним інтелектом, які представляють собою набори організованих та зв’язаних між собою скриптів (в рамках даного проекту використовувалась оболонка Python). Крім цього було розроблено власні скрипти для виконання розпізнавання та візуалізації результатів.

Апаратні засоби

В рамках роботи дипломного проекту, було обрано використовувати центральний процесор для вирішення основних задач системи.

Говорячи про потенційні модифікації системи з метою підвищення ефективності та швидкості її роботи, було обрано наступні відповідності процесорів та задач:

1. Задачі класифікації – центральний процесор.
2. Графічна обробка – графічний процесор.
3. Обробка зображення нейронною мережею – FPGA процесор.

3. РОЗРОБЛЕННЯ АЛГОРИТМІВ ТА МОДУЛІВ

3.1. Загальний опис ядра системи

Ядро системи складається з двох основних модулів, які взаємодіють один з одним та із зовнішніми підсистемами (база даних та сервер) впродовж всього циклу роботи (див. рис. 4):

- модулю контролю;
- модулю аналізу.

З пристрою (камери) дані надсилаються на мікроконтролер, в якості якого було обрано Raspberry Pi 3 Model B. Raspberry Pi – це серія невеликих одноплатних комп'ютерів, розроблених Фондом Raspberry Pi в Англії для сприяння викладанню базової інформатики в школах та університетах.

Рис. 4. Діаграма варіантів використання модулів ядра системи

Модель Raspberry Pi 3 B – це найперша модель Raspberry Pi третього покоління. Має наступні характеристики:

- Quad Core 1.2GHz Broadcom BCM2837 64-бітний процесор;
- 1 Гб оперативної пам'яті;
- порт камери CSI для підключення камери Raspberry Pi;
- Micro SD порт для завантаження операційної системи та зберігання даних;
- тощо.

Мікроконтролер, виконує початкову обробку даних в рамках роботи із модулем контролю, отримуючи зображення з камери та аналізуючи його кожні 4 секунди, налаштовуючи при цьому з’єднання із зовнішніми підсистемами, надсилаючи результат модулю аналізу. Обробивши ці дані, останній отримує кінцевий результат обробки в наступному форматі: зображення, що містить у собі результати аналізу (ідентифіковані та позиціоновані влучення), xml-файл, що містить деталізовану інформацію про кожне з ідентифікованих влучень.

Далі модуль аналізу одночасно взаємодіє з базою даних, а також із сервером, на якому вона розгорнута. В базу даних відправляється запит на виклик процедури, яка приймає в якості параметрів інформацію про влучення та додає її в якості запису до відповідної таблиці [18]. Паралельно, безпосередньо серверу надається доступ до директорії, в якій зберігається зображення, що є результатом обробки. Сервер копіює це зображення для подальшого виведення в графічному інтерфейсі користувача [19].

3.2. Архітектура ядра системи

«Предиктором» в рамках системи є Motion, структуру якого було описано вище. Розглянемо структуру «Коректора».

«Коректор» є модулем, написаний на python, що виконує наступні функції:

- розпізнавання влучень у мішені та їх позиціонування;
- маркерування місць влучень у вигляді лейблів, що обмежують місце влучення та відображають порядковий номер;
- відправлення даних про влучення у базу даних, надання доступу до результуючого зображення (що включає в себе візуалізовані маркерування).

Модуль складається з наступних підмодулів:

- BaseObject.py – зберігає в собі наслідувані поля-списки, які зберігають інформацію про дані та операції класу. Надають доступ до оператора ‘[]’, за рахунок використання якого можливо отримати доступ до відповідних даних або вказівників на методи, що знаходять всередині списків.
- DetectionResult.py – модуль, що зберігає в собі клас DetectionResult, функціями якого є збереження результатів розпізнавання як атрибутів та надання доступу до них зовнішнім класам для здійснення відповідних маніпуляцій (візуалізація, виведення тощо).
- Detector.py – модуль, що зберігає в собі клас AbstractDetector, функціями якого є здійснення розпізнавання певного об’єкту (або змін в об’єкті). Є абстрактною сутністю та надає доступ до внутрішніх операцій та атрибутів класам-нащадкам.
- BaseConfigManager.py – модуль, що зберігає в собі клас BaseConfigManager, функціями якого є організація області пам’яті, доступ до якої надається іншим класам шляхом використання патерну Decorator. Доступ здійснюється за допомогою використання операції отримання доступу за індексом “[]”, результатом виконання якої є місце в пам’яті із відповідними даними.

- Saver.py – модуль, що зберігає в собі клас Saver, функціями якого є виконання збереження даних, які були отримані в результаті розпізнавання.
- Visualizer.py – модуль, що зберігає в собі клас AbstractVisualizer, функціями якого є візуалізація розмітки, яка позиціонує зміни всередині об’єкту.
- DBConnector.py – модуль, що зберігає в собі клас DBConnector, функціями якого є здійснення комунікації із базою даних, яка включає в себе: підключення до бази даних на основі наданих параметрів, передача даних про влучення у форматі (координати (x, y), час (формат DateTime), індекс мішені).
- Detect.py – основний модуль, що містить в собі точку входу та код основного процесу виконання програми (Додаток 4).

Для кожного з модулів, що зберігають в собі класи наявні конфігураційні файли із даними, які є необхідними для коректної роботи внутрішніх алгоритмів:

- DetectionResultConfig.py – конфігураційний файл, що зберігає в собі значення параметрів для роботи алгоритмів збереження даних розпізнавання.
- DetectorConfig.py – конфігураційний файл, що зберігає в собі значення параметрів, що є необхідними для роботи базових операцій алгоритмів розпізнавання.
- BaseConfigManagerConfig.py – конфігураційний файл, що зберігає в собі значення параметрів для роботи функцій організації області пам’яті для збереження параметрів та її подальшої розмітки.
- SaverConfig.py – конфігураційний файл, що зберігає в собі дані для роботи функцій збереження результатів розпізнавання.

- VisualizerConfig.py – конфігураційний файл, що зберігає в собі значення параметрів, які необхідні для виконання збереження результатів збереження.
- GeneralConfig.py – конфігураційний файл, який зберігає в собі загальні налаштування для системи.

3.3. Модуль контролю

Модуль контролю – модуль, що забезпечує коректну взаємодію апаратної частини (мікроконтролери, камери тощо) із алгоритмічними процесами, що відбуваються всередині системи. Основною функцією даного модулю є організація роботи на основі низькорівневого програмування та реалізації моделі міжпроцесної взаємодії для обробки даних та сигналів, що надходять з пристроїв. Включає в себе виконання ролі «Предиктора» у моделі «Предиктор-Коректор».

В якості даного модулю було обрано систему моніторингу відео-сигналів, яка виконує детекцію змін на сцені, яку знімає камера (іншими словами, дана система виконує моніторинг рухів), яка називається Motion. Дана система розроблена в рамках GNU General Public License версії №2 або пізніше, що дозволяє її використання та розповсюдження. До системи надається файл copyright-у, в якому прописана відповідна ліцензія. Крім цього додається детальна документація, яка вміщує в себе як інформацію про процес інсталяції, так і опис всіх внутрішніх модулів системи та їх змісту.

Дана система написана засобами мови С і реалізовує у собі наступні основні функції:

- запуск модулю у різних режимах (як основний процес або як демон) та налаштування параметрів реєстрації повідомлень про її стан;
- налаштування параметрів роботи відео-пристрою (або пристроїв), захоплення яких здійснюється, за допомогою v4l2loopback-модулю.

Можливе захоплення пристрою, підключення якого здійснюється як локально, так і за веб-інтерфейсом (через URL);

- налаштування протоколів обміну даними з пристроями як з точки зору їх обробки, так і з точки зору режиму, в якому здійснюється їх відправлення із подальшим налаштуванням параметрів згідно яких буде здійснюватись конвертації зображення сцени (роздільна здатність, кількість кадрів в секунду, обертання зображення тощо);
- організація базової передобробки зображення сцени (встановлення порогу дозволеного шуму на зображенні, накладання маски, обробка змін світла тощо);
- організація вихідних даних в якості зображення, відео, відео-потоку із можливим налаштуванням протоколів обміну даними, портів, виду представлення IP-адреси тощо;
- налаштована конфігурація для роботи з наступними базами даних: mysql, postgresql, sqllite3, - та подальшою можливістю взаємодії з їх екземплярами, що можуть бути розташовані на віддалених вузлах.

Встановлення модуля здійснюється шляхом його компіляції з вихідного коду[1]. В репозитарії проекту знаходяться відповідні файли конфігурації та збірки, що дозволяють здійснити його компіляцію та подальшу інсталяцію на машину.

Основні підмодулі:

1. motion.c – модуль, що зберігає в собі точку входу програми, а також методи, які виконують ініціалізацію базових параметрів: організація пам’яті, багатопоточної обробки, взаємодії з пристроями, базами даних, налаштування основних етапів роботи системи тощо.

[1] Вихідний код системи можна знайти за посиланням - https://github.com/Motion-Project/motion.

Зміни були внесені в наступний методи даного модулю:

static void dbse_global_init(void) – ініціалізація зв’язків із базами даних, параметри яких були вказані всередині конфігураційного файлу. Даний метод доповнений функцією перевірки можливості встановлення зв’язку із базою даних, що здійснює обробку інформації про влучення, а також надає користувачеві графічний інтерфейс. Частину коду методу, а також код функції наведено у додатку 2.

2. alg.c – модуль, що зберігає в собі всі алгоритмічні функції, які використовуються для виконання передобробки зображення. Специфіка алгоритмів є доволі варіативною, реалізовано по декілька версій алгоритмів для: виконання локації змін, обробки шуму, базових морфологічних операцій, накладання маски, визначення різниці між зображеннями, обробки світла тощо.

   Основними методами є:

   - int alg_diff_standard(struct context *cnt, unsigned char *new) – реалізація алгоритму, який розраховує різницю між двома зображеннями. В якості параметрів контекст cnt, а також зображення, представлене масивом символів. Розрахунки виконуються при використанні MMX-регістрів та пайпів (Додаток 2).
   - void alg_noise_tune(struct context *cnt, unsigned char *new) – метод, що виконує обробку шумових ефектів у зображенні (Додаток 2).

3. mmx.h – модуль, що зберігає в собі набір макросів для здійснення маніпуляцій з mmx-регістрами. В даному модулі реалізовано більше двохсот макросів, кожен з яких представляє операцію над даними регістру (Додаток 2).

3.4. Модуль аналізу

Загальна інформація

Модуль аналізу – модуль, в якому реалізована обробка отриманого зображення з метою знайдення та позиціонування змін в об’єкті (влучень в мішень). В рамках модулю дана задача вирішується за допомогою використання штучного інтелекту. Виконує роль «Коректора» у моделі «Предиктор-Коректор».

Даний модуль було розроблено засобами бібліотеки Tensorflow (версії 1.9) та створеного для неї Tensorflow Detection API, що надає розробнику можливості моделювання, навчання та використання інструментів штучного інтелекту, а саме – нейронних мереж. В даному випадку нейронна мережа, яку використовує система, «навчалась» на зображеннях мішеней, на яких були відповідним чином промаркіровані влучення.

Бібліотека Tensorflow. Google Object Detection API

TensorFlow – відкрита бібліотека для машинного навчання, розроблена компанією Google для вирішення завдань побудови і тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття. Застосовується як для досліджень, так і для розробки власних продуктів Google. Основний API для роботи з бібліотекою реалізований для Python, також існують реалізації для C++, Haskell, Java, Go і Swift.

Дана бібліотека є побудованою на основі тензорного процесору, що відноситься до класу нейронних процесорів та за своєю суттю є інтегральною схемою, яка була розроблена корпорацією Google. Пристрій реалізований як матричний помножувач для 8-розрядних чисел, керований CISC-інструкціями центрального процесора. Тактова частота становить 700 МГц і має теплову розрахункову потужність 28-40 Вт. Оснащений 28 Мбайт вбудованої оперативної пам'яті і 4 Мбайт 32-розрядних акумуляторів, що

накопичують результати в масивах з 8-бітних множників, організованих в матрицю розміром 256 × 256. Інструкції пристрої передають дані на вузол або отримують їх з нього, виконують матричні множення або згортки [20]. В такт може проводитися 65536 множень на кожній матриці; в секунду – до 92 трильйонів [21].

TensorFlow Object Detection API є фреймворком із відкритим вихідним кодом, побудованим на основі TensorFlow, що дозволяє легко створювати, тренувати і розгортати моделі виявлення об'єктів.

Процес навчання нейронної мережі

В процесі навчання нейронної мережі можливо виділити три основні етапи:

1. Підготовка наборів даних. Дані, на основі яких здійснюється навчання, готуються у вигляді трьох наборів – навчального набору (training set) [22], затверджуваного набору (validation/evaluation dataset) та тестового набору (test dataset).
2. Навчальний набір даних є множиною прикладів, що використовуються для навчання (тобто, для налаштування вагів нейронної мережі таким чином, щоб кінцевий результат оцінки відповідав очікуваному).
3. Набір даних для затвердження використовується для створення об’єктивної оцінки моделі, що відповідає даному навчальному наборі в процесі налаштування гіперпараметрів моделі [23]. В машинному навчанні гіпермараметр – це параметр, значення якого встановлено ще до того, як починається процес навчання. Прикладами гіперпараметрів можуть бути: кількість листків (або глибина) дерева, кількість прихованих шарів тощо.
4. Тестовий набір є множиною прикладів, що використовується для створення «об’єктивної» оцінки кінцевої моделі, що відповідає даному навчальному набору даних.

В рамках роботи з Object Detetction API дані мають бути подані у вигляді файлу з розширенням .record, який зберігає у собі модель набору даних, який може створюватись різними шляхами (в даній системі розглядається варіант створення .record-набору на основі Pascal VOC Dataset) та у подальшому використовуватись для навчання (маркерування даних для створення наборів виконувалось за допомогою відкритої утиліти LabelImg [24]).

Формування набору здійснюється за рахунок виклику скрипту із відповідними параметрами: директорії із даними, вихідного шляху, шляху до файлу-розмітки (Додаток 2). На найвищому рівні абстракції, конфігураційний файл розділений на наступні етапи:

1. Конфігурація моделі. Визначає, який тип моделі буде підготовлено (тобто мета-архітектура, екстрактор ознак).
2. Конфігурація навчання, що визначає, які параметри слід використовувати для підготовки параметрів моделі (наприклад, параметри SGD [25], які включають значення попередньої обробки вхідних даних і значення ініціалізації екстрактора функцій).
3. Конфігурація оцінки, що визначає, який набір метрик буде передано розрахунку для оцінки.
4. Вихідна навчальна конфігурація, що визначає, на якому наборі даних треба навчити модель.
5. Вихідна конфігурація оцінки, що визначає, який набір даних модель буде оцінювати. Зазвичай він має відрізнятися від набору вихідних навчальних даних.

   Існує велика кількість параметрів моделі для налаштування. Найоптимальніший варіант налаштування залежить від застосунку в цілому та специфіки задач, які він вирішує (наприклад, R-CNN моделі краще підходять для випадків, коли висока точність є бажаною і затримка має нижчий пріоритет, і навпаки, якщо час

обробки є найважливішим фактором, рекомендується використовувати моделі SSD).

6. Навчання моделі. Для виконання навчання, необхідна структура, що включає в себе не тільки набори даних. Одним з елементів даної структури є тренувальна пайплайн-конфігурація. Object Detection API використовує файли protobuf для налаштування процесу навчання та оцінки. Схема для навчального пайплайну може бути знайдена в object_detection/protos/pipeline.proto. Даний конфігураційний файл має певний скелет, що містить у собі параметри, необхідні для організації процесу навчання (Додаток 2) [26].

   В рамках розробки системи, в якості нейронної мережі був використаний відповідний чекпоінт, отриманий з моделі ssd_mobilenet_v1_coco та її конфігураційний файл, в якому були змінені шляхи як до папки, в якій зберігається модель, так і до місць, де зберігаються набори даних.

   Другим елементом структури є файл розмітки з рорширенням .pbtxt (Додаток 2), який слугує «картою» для нейронної мережі у відповідності з якою відбувається визначення списку об’єктів, згідно з яким необхідно виконувати ідентифікацію та локалізацію об’єкту.

   Фінальним етапом є запуск скрипту якому в якості параметрів надаються наступні параметри: флаг логування, шлях до директорії, в якій необхідно зберігати результат, шлях до конфігураційного файлу (Додаток 2).

7. Експорт графу виводу [27]; підготовка до розпізнавання. Після закінчення процесу тренування, необхідно отримати статичний граф виводу отриманої моделі на основі вибраного чекпоінту. В результаті даної операції отримаємо наступні елементи:
   - збережену модель та папку, що зберігає формат моделі, на основі якої вона будувалась;
   - статичний граф виводу;
   - чекпоінт model.ckpt.*, який можливо використовувати для експорту;
   - файл checkpoint, який формується під час процесу навчання і оновлюється з кожним збереженням моделі. З нього можливе відновлення чекпоінтів, які були зроблені протягу всього процесу;
   - конфігураційний пайплайн-файл для моделі, що була отримана.

   На основі отриманих даних в подальшому буде можливе здійснення процесу розпізнавання (Додаток 2).
8. Процес розпізнавання. Останнім кроком є безпосереднє виконання розпізнавання на основі отриманих даних. Для даної задачі було сформовано набір скриптів у вигляді окремої програми. Параметрами, що приймає дана програма є результат операції, що виокремлює граф виводу (структура файлів, яка повністю описує кінцеву модель), файл-розмітку та тестові файли (у випадку системи – зображення мішені) (Додаток 4).

В результаті отримуємо xml-файл, що містить у собі розмітку, сформованою нейронною мережею для даного зображення та промаркероване зображення (див. рис. 3, 4).

Рис. 3. Результат розпізнавання трьох влучень

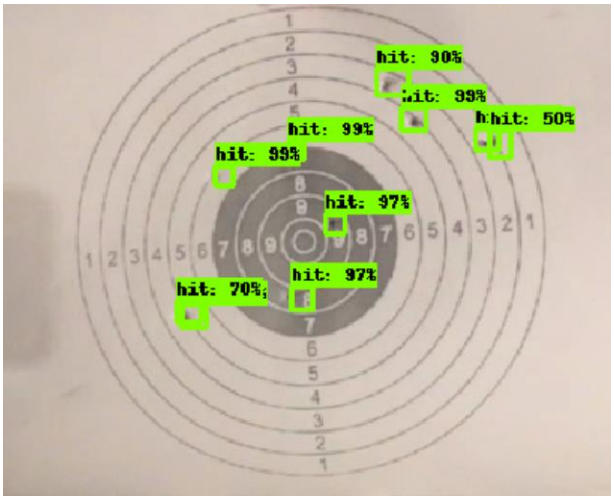


Рис. 4. Результат розпізнавання семи влучень

3.5. Висновки до розділу

В рамках даного розділу було розглянуто основні модулі системи та деталізовано як їх наповнення, так і засоби, що застосовувались при розробці.

Було наведено процеси, що відбуваються всередині модулів, що реалізовують в собі модель «Предиктор-Коректор», розглянуті основні алгоритми роботи та наведено приклади коду.

Крім цього, було розглянуто процес навчання нейронної мережі, а також її використання безпосередньо в рамках системи на прикладі результатів розпізнавання.

4. АНАЛІЗ РОЗРОБЛЕНОГО ЯДРА СИСТЕМИ ДИНАМІЧНОГО ВИЗНАЧЕННЯ ВЛУЧЕНЬ У МІШЕНЬ

4.1. Практичне застосування системи та її алгоритмів, закладених у ядрі

Стрілецький спорт за останні роки збільшив і продовжує збільшувати свою популярність. За даними досліджень на 2020 рік ринок стрілецького спортивного обладнання буде становити близько 1 млрд. доларів, тим не менш більше 70% споживачів відповідного сегменту використовують ручні збір та аналіз даних [28]. Також автоматизовані системи не використовуються навіть на міжнародних змаганнях. Крім цього, дана система надає можливість підвищення рівня стрільби не тільки сегменту спортивної стрільби, а й воєнним.

Говорячи безпосередньо про алгоритми, закладені всередині ядра, варто відмітити, що їх база може використовуватись в інших сферах, які потребують вирішення тих чи інших завдань, зв’язаних з позиціонуванням. Основними з них є астрономія та медицина.

Астрономічні дослідження вимагають постійного стеження за об’єктом, збору даних про нього та їх обробки. Використання автоматизованих алгоритмів позиціонування стає можливим за рахунок постійних розробок автоматизованих телескопів різних розмірів та призначення, а постійна необхідність слідкування за небесними тілами робить їх використання ресурсовитратним з точки зору доступного часу та наявної людської сили.

В рамках медицини швидко іде розвиток високоточних мікроскопів та медичних апаратів, що працюють як в межах автоматизованих комплексів, так і окремо. Тим не менш, внаслідок мутацій вірусів через активне використання людством антибіотиків, і прилади, і людина можуть робити хибні висновки з приводу того чи іншого захворювання, особливо якщо

вердикт базується на складі крові, аналіз якої здійснювався. Використання даної системи може бути корисним з метою підрахунку кількості клітин у краплині крові шляхом виконання позиціонування, на основі чого можливе виконання корекції результатів, що надаються спеціалізованим апаратом.

Окрім цього, даний підхід можливо використовувати у різноманітних системах моніторингу змін на сцені, що включає в себе як процес слідкування за рухом певного об’єкту, так і аналіз та передбачення траєкторії, напрямку та швидкості руху.

Розроблюване ядро надає можливість свого використання в рамках різних сфер за рахунок роботи моделі «Предиктор-Коректор», яка реалізована таким чином, що «Предиктор» є платформою, яка не змінює свою структуру та підходи до обробки даних і методів регулювання та управління системою. В даному випадку зміни будуть вноситись до конфігураційних даних, але не до самої платформи. «Коректор», у свою чергу, є частиною, яка буде набувати змін, але не дуже значних. Як загальний алгоритм роботи, так і алгоритм взаємодії із «Коректором» та зовнішніми підсистемами залишається тим самим, зміни вносяться лише до нейронної мережі в залежності від задач, які необхідно вирішувати системі.

4.2. Порівняння з традиційними засобами розроблення

Основні підходи до технологій розробки, що присутні у сучасних системах динамічного позиціонування:

1. Організація комплексу пристроїв, що здійснює вимірювання великої кількості параметрів. Даний підхід є популярним в системах позиціонування судів та інших масштабних об’єктів на місцевості. Пристрої здійснюють вимірювання таких параметрів, як: вітер, течія, відстані від певних об’єктів тощо. В результаті обробки цих даних можливо отримати необхідну позиції, в якій знаходиться досліджуваний об’єкт, а також позицію, в яку він має

перейти. Даний підхід достатньо важко застосувати в рамках вирішення задачі позиціонування влучень в мішень.

2. Організація процесу моніторингу об’єкту. Даний підхід широко використовується в охоронних системах та стрілецькому спорті. В якості пристрою, що здійснює моніторинг, використовується камера. При цьому наявність певних автоматичних рішень є доволі рідкісним фактором, в результаті чого людині необхідно самостійно здійснювати слідкування за сценою. В рамках стрілецьких тренувань, люди іноді використовують камери для здійснення слідкування за мішенню із метою підтвердження наявності або відсутності влучення. Тим не менш, даний спосіб моніторингу не є дуже ефективним.
3. Використання лазерних указок при стрільбі. Доволі популярним рішенням є приєднання до зброї лазерних указок, що пускають свій луч разом із пострілом. Відносно параметрів лучу, який пускає указка, здійснюється моделювання в результаті якого визначається місце, в якому має бути влучення. Дане рішення є відносно легким у реалізації, але втрачає в точності та наявності додаткових датчиків, а рахунок яких можна відстежувати, яким чином пішов луч. Крім цього, можливі проблеми з точністю вимірювань через накопичення похибок при обчисленні, а також можливих зовнішніх факторах впливу.
4. Використання високоточних лазерних пристроїв. Даний підхід використовується у деяких існуючих системах динамічного позиціонування влучень. Основними позитивними характеристиками даних рішень є висока точність вимірювань та, як ітог, висока точність результатів пошуку влучень в мішені. При цьому, процес ідентифікації є достатньо швидким [29] для використання системи на тренуваннях. Тим не менш, рішення має

очевидні недоліки: процес переміщення системи є доволі трудомістким через громіздкість конструкції та самих лазерів; одночасно можливий аналіз однієї мішені, за необхідності виконання аналізу декількох мішеней, лазер потребує відключення, зміни напрямлення лучу та повторного підключення; використання лазерних технологій є доволі дорогим підходом, внаслідок чого собівартість системи піднімається в рази, що потенційно тягне за собою зменшення кількості клієнтів.

Основним недоліком рішень, вказаних вище, залишається неможливість взаємодії даної системи у неприйнятних погодних умовах через відсутність моделі, що включає в себе характеристики системи динамічного позиціонування Класів 2 та 3. В результаті це тягне за собою як відсутність можливості роботи з фільтрацією, так і отримання автоматичного підтвердження, чи справді у мішені з'явилось влучення. Саме тому в розроблюваному ядрі передбачена робота моделі «Предиктор-Коректор», що надає всій системі характеристики Класів 2 та 3, дозволяючи при цьому використовувати мінімальну кількість технічного обладнання для її подальшого розгортання.

4.3. Оцінка продуктивності системи

Оцінка продуктивності є нелегкою, але необхідною задачею як при дослідженні, так і при проектування або розробці нової системи. Існує багато варіантів пошуку даної величини, але всі вони базуються на тому, що кожний з системних компонентів системи здійснює свій певний вплив на кінцевий результат. Інакше кажучи, якби було доступне рівняння, яке описує продуктивність системи, кожному з її елементів відповідала певна змінна в цьому рівнянні.

Розглянемо наступні основні характеристики, які разом формують представлення про продуктивність системи:

- Процесорне навантаження. Дана характеристика визначається ступенем навантаження центрального процесора під час роботи системи. Для визначення навантаження, що здійснюється на процесор системою, достатньо оцінити наступні величини: процент часу, що витрачається на виконання користувацьких задач (usr), процент системного часу (sys), процент часу, який витрачається на очікування вводу/виводу (wio), процент часу простою центрального процесора (idle). При цьому сумарне навантаження має складати 100%: $usr + wio + wio + idle = 100\%$. Для системи, що функціонує в нормальних умовах, зазвичай має виконуватись наступне [30]: $usr + wio + sys \leq 80\%$, $wio \leq 20\%$, $sys \leq 70\%$. Показник sys може змінюватися широких межах, від типових 10%-30%, до 50% і більше. Це пов'язано з тим, що деякі мережеві сервіси (наприклад, NFS) є частиною ядра, і будь-яка активність NFS відноситься не до показника користувальницької завантаження, а до показника sys, отже, його не можна використовувати для оцінки продуктивності. Показник завантаження wio не повинен перевищувати 30%, в іншому випадку можна з упевненістю говорити про низьку пропускну здатність (а отже і продуктивність) системи введення-виведення. Сумарний відсоток утилізації в нормально завантаженої системі не повинен перевищувати 70%-80%, так як при збільшенні завантаження центральний процесор збільшується чергу процесів, які очікують на обслуговування, а отже і час їх виконання.

  В таблиці 1 наведені значення величин *usr, wio, sys, idle* та значення сумарного навантаження під час роботи ядра системи.

Таблиця 1

Значення величин, що оцінюють процесорне навантаження мікроконтролера під час роботи ядра системи

Величина	Значення, %
usr	55
wio	20
sys	5
idle	20
usr + wio + sys + idle	100

- Пам’ять. Для оцінки даної величини будемо використовувати значення вільної, зайнятої пам’яті, а також пам’яті, що використовується операційною системою під час двох станів роботи системи: стан обробки даних, стан очікування. В результаті маємо наступні значення (див. табл. 2):

Таблиця 2

Значення величин, що дозволяють оцінити використання пам'яті мікроконтролера під час роботи системи

Величина	Значення, %	
	Під час обробки даних	Під час очікування на дані
Зайнята пам’ять	80	25
Вільна пам’ять	10	50
Пам’ять, використовувана ОС	10	25

4.4. Напрямки подальшого вдосконалення

Вдосконалення моделей обробки

Одним з варіантів вдосконалення системи (який є необхідністю за умови просування системи як продукту) є модернізація моделі «Предиктор-Коректор» шляхом надання їй більш комплексного наповнення.

На даний момент алгоритм «Предиктора» цілком являє собою роботу Motion, в результаті вміщуючи в себе метод, за яким Motion здійснює порівняння зображень для ідентифікації руху на сцені. Даний алгоритм працює добре в умовах, коли необхідно позиціонувати масштабні зміни, але є дуже неоптимальним у використанні для вирішення задач позиціонування малих змін; сам алгоритм здійснює порівняння двох зображень шляхом повного перебору пікселів зображення та підрахунку тих, що змінились. Результатом є збільшення області інтересу і, як наслідок, кількості операцій, які необхідно виконати програмі для знаходження відмінностей.

Модернізований алгоритм «Предиктора» має виглядати таким чином, що під час порівняння зображень система:

- зменшує загальну область, в рамках якої буде виконуватись подальша обробка шляхом позиціонування на сцені самої мішені, обрізаючи область, що її оточує;
- зведення зображень до одного шаблону шляхом математичних перетворень;
- виконання різниці між двома зображеннями, в результаті якої отримуємо тільки зміни, що відбулись.

Після виконання даного алгоритму результати надаються нейронній мережі, яка, у свою чергу, також буде змінена. Дана нейронна мережа буде навчена розпізнавати та позиціонувати влучення не як об’єкт всередині мішені, а я об’єкт всередині результату різниці. Відповідно, робота «Коректора» також змінюється.

Вдосконалення апаратної частини

В рамках апаратної частини вдосконалення можливе відносно роботи із різними типами процесорів. Розглядаючи апаратні частини систем динамічного позиціонування, варто виділити наступні проблеми, які часто виникають:

- Неефективна організація системи процесорів. Часто один і той самий тип процесорів використовується для різних задач, що містять ті, до яких він не пристосований. Прикладом можна вважати використання центрального процесора, робота з яким є найпопулярнішою, для такої задачі, як графічна обробка.
- Відсутність оптимізованих адаптаційних бібліотек комп'ютерного зору для різних типів процесорів (певні види процесорів вимагають безпосереднього перепрограмування для виконання подальшої обробки основними бібліотеками, що використовують системи комп'ютерного зору).

Відповідно, існує потреба в системі, яка б не тільки була адаптивною у контексті розмірів змін, які необхідно ідентифікувати та позиціонувати, а й містила у собі коректну апаратну платформу, кожний модуль якої б виконував відповідну його типу задачу.

4.5. Висновки до розділу

В рамках даного розділу було частково проаналізовано результати розробки ядра системи динамічного позиціонування влучень у мішень під час виконання стрільб.

Було розглянуто практичне застосування системи, що включило в себе інформацію про бізнес-складову та степінь необхідності розробки даної системи у відповідності до проблем сегменту та ніші, які вона вирішує. Крім цього, було наведено приклад галузь, в рамках яких технології, закладені всередині даної системи можливо використовувати з

метою вирішення завдань позиціонування тих чи інших об’єктів або змін в їх стані (медицина, астрономія тощо).

Було проведено порівняння із підходами, що використовуються в подібних системах, які вирішують задачі позиціонування і наведено переваги та недоліки кожної з них, а також засоби, що було закладено в рамках даного дипломного проекту для вирішення відповідних проблем та виправлення недоліків із метою підвищення рівня ефективності та оптимізації роботи.

Також було виконано оцінку продуктивності системи з точки зору процесорного навантаження та пам’яті, що використовується під час роботи (як системою, так і операційною системою в цілому). Було оцінено наступні величини: процент часу, що витрачається на виконання користувацьких задач (*usr*), процент системного часу (*sys*), процент часу, який витрачається на очікування вводу/виводу (*wio*), процент часу простою центрального процесора (*idle*); зайняту та вільну оперативну пам’ять під час обробки даних та очікування на їх введення/виведення, а також процент оперативної пам’яті, яку використовує операційна система.

Було запропоновано підходи потенційного вдосконалення системи, серед яких:

- Вдосконалення моделі обробки, в рамках якої «Предиктор» буде реалізовувати в собі більш складний та ефективний алгоритм, що дозволить формувати область змін не як суцільний масштабний регіон, а як невеликі регіони інтересу, які мають бути передані нейронній мережі на подальшій аналіз (таким чином, загальний час роботи буде зменшено).
- Вдосконалення апаратної частини, що веде до суттєвого прискорення системи через розподілення задач різного призначення між відповідними процесорами різних типів.

ВИСНОВКИ

Використання систем динамічного позиціонування є поширеним явищем у певних сферах нашого життя. У більшості випадків вони використовуються із метою вирішення проблем позиціонування того чи іншого об’єкту або змін всередині даних об’єктів шляхом використання автоматизації аналізу їх параметрів, на основі яких здійснюється формулювання задачі позиціонування.

Існує декілька класів систем позиціонування:

1. Клас 0 – система, позиціонування об’єкту дослідження якої відбувається вручну, а аналіз руху – автоматично за умов максимального впливу зовнішніх чинників.
2. Клас 1 – система, позиціонування об’єкту дослідження якої, а також аналіз його руху можуть відбуватись як вручну, так і автоматично за умов максимального впливу зовнішніх чинників.
3. Клас 2 – система, позиціонування об’єкту дослідження якої, а також аналіз його руху можуть відбуватись як вручну, так і автоматично за умов максимального впливу зовнішніх чинників та при наявності несправностей при паралельному використанні двох екземплярів системи.
4. Клас 3 – система, позиціонування об’єкту дослідження якої, а також аналіз його руху можуть відбуватись як вручну, так і автоматично за умов максимального впливу зовнішніх чинників та при наявності несправностей при паралельному використанні принаймні двох екземплярів системи, для кожного з яких наявна можливість відновлення з резервної копії.

Тим не менш, більшість систем, які можна віднести до будь-якого з класів має певні проблеми з наступних:

1. Системи з несуттєвими розмірами об’єктів відповідно до розмірів самої системи.
2. Системи з обмеженням по масштабуванню та релевантності закладених математичних моделей.
3. Системи з обмеженнями по апаратним засобам реалізації.

В результаті було змодельовано та розроблено систему динамічного позиціонування змін в стані об’єкту, яка б вирішувала або потенційно вирішити на основі отриманої розробки вказані проблеми. Задача системи – пошук влучень у мішень під час стрільби (в даному випадку об’єктом є мішень, змінами, які необхідно ідентифікувати та позиціонувати – влучення). Система складається з трьох основних компонентів: камера, мікроконтролер управління, сервер, який зберігає в собі базу даних та веб-сервер. Загальний алгоритм роботи даної системи наступний:

1. Камера здійснивши зйомку мішені, відправляє дані на мікроконтролер.
2. Мікроконтролер виконує аналіз даних.
3. Результати аналізу потрапляють на сервер та у базу даних.
4. Після отримання здійснюється відображення результатів аналізу в графічному інтерфейсі користувача.

В рамках даного дипломного проекту здійснювалась розроблення ядра системи динамічного позиціонування влучень у мішень із залученням елементів наступних технологій: комп’ютерний зір, штучний інтелект. В основу даної розробки було покладено модель «Предиктор-Коректор» із потенційною можливістю покращення системи або її модифікування. Система складається з двох основних модулів:

1. Модуль контролю – забезпечує коректну взаємодію апаратної частини (мікроконтролери, камери тощо) із алгоритмічними

процесами, що відбуваються всередині системи. Виконує «Предиктора» у моделі «Предиктор-Коректор».

2. Модуль аналізу – модуль, в якому реалізована обробка отриманого зображення з метою знайдення та позиціонування змін в об’єкті (влучень в мішень). Виконує роль «Коректора» у моделі «Предиктор-Коректор».

Окрім безпосередньої розробки було: розглянуто шляхи вдосконалення як системи в цілому, так і її внутрішніх моделей та алгоритмів; проведено базовий бізнес-аналіз перспектив системи в сфері систем автоматизації процесів спортивної та воєнної стрільби; проаналізовано можливості використання технології позиціонування, яку було покладено в основу розробки, для вирішення задач в інших сферах (медицина, астрономія тощо).

СПИСОК ЛІТЕРАТУРИ

1. Кр. Час, Р. Гарсія. Introduction to ship dynamic positioning systems – Іспанія: Сантандер, 2008 – №V (1) – Journal of Maritime Research, ISSN.
2. Фр. Абулуде, Ак. Акіньїнка. Global Positioning System and It's Wide Applications – 2015 – №9 (1) – Continental J. Information Technology, ISSN 10.5707/cjit.2015.9.1.22.32.
3. Х. Маннай. Location and Positioning Systems: Performance and comparison / Х. Манная, Н. Бенадйоссеф, М. Махут, Х. Уренья – Туніс: Хаммамет, 2016 – ISBN: 978-1-5090-1055-4.
4. Offshore Engineering. DP Position Reference Systems – Отримано з Offshore Engineering: https://offshoreengineering.com/education/dynamic-positioning-dp/dp-position-reference-systemsю
5. М. Сонка. Image Processing, Analysis, and Machine Vision / М. Сонка, В. Лавач, Р. Бойль – 2008 – ISBN 978-0-495-08252-1.
6. Ст. Рассел, П. Норвиг. Artificial Intelligence: A Modern Approach – 2010 – №3 – Prentice Hall ISBN 9780136042594.
7. М. Мері. Foundations of Machine Learning / М. Мері, А. Ростамізаде, А. Тальвалькар – 2012 – The MIT Press ISBN 9780262018258.
8. Дж. Хінтон; С. Терренс. Unsupervised Learning: Foundations of Neural Computation – 1999 – MIT Press. ISBN 978-0262581684.
9. А. Каплан. Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence / А. Каплан, М. Хаенлейн – 2018 – Business Horizons – № 62(1) – С. 1-2.
10. Дж. Хуанг. Speed/accuracy trade-offs for modern convolutional object detectors / Дж. Хуанг, В. Ратод, Чен Сан, М. Жу, Ан. Кораттікара, А. Фаті, Іан Ашшер, Зб. Вонжа, Янг Сонг, С. Гуадаррама, К. Мерфі – 2017 – США: Гонолулу – IEEE 10.1109/CVPR.2017.351.

11. А. Крижевскі. ImageNet classification with deep convolutional neural networks / А. Крижевскі, І. Сутскевер, Г. Хінтон – 2012 – NIPS.
12. Р. Гіршик. Rich feature hierarchies for accurate object detection and semantic segmentation / Р. Гіршик, Дж. Донае, Тр. Даррел, Дж. Малік – США: Берклі; ICSI, 2014 – Конференція IEEE з теми: «Комп'ютерний зір та розпізнавання шаблонів» – С. 580-587.
13. С. Рен. Faster r-cnn: Towards real-time object detection with region proposal networks / С. Рен, К. Хі, Р. Гіршик, Дж. Сан – 2015 – Advances in neural information processing systems – С. 91-99.
14. В. Ліу. SSD: Single Shot MultiBox Detector / В. Ліу, Др. Ангелов, Д. Ерхан, К. Зежеді, Ск. Рід, Ченг-Янг Фу, А. Берг. – 2016 – США: Університет Північної Кароліни, Кароліна; Zoox Inc, Каліфорнія; Google Inc., Каліфорнія; Університет Мічігану, Енн-Арбор – ECCV 2016 10.1007/978-3-319-46448-0_2.
15. С. Аі Гуо-Хьянг. A Positioning System based on Communication Satellites and the Chinese Area Positioning System (CAPS) – 2008 – Chinese Journal of Astronomy and Astrophysics.
16. Х. Клатем Аль Нуаімі. A survey of indoor positioning systems and algorithms – Міжнародна конференція з приводу інновацій в Інформаційних технологіях, Абу-Даби – IEEE.
17. Т. Пітерс. PEP 20 – The Zen of Python – 2004 – Отримано з www.python.org: https://www.python.org/dev/peps/pep-0020/
18. Д. Охримчук. Проектування інтерфейсної частини системи динамічного позиціонування змін у стані об'єкту. Розроблення бази даних – 2019 – С. 20-30.
19. Д. Охримчук. Проектування інтерфейсної частини системи динамічного позиціонування змін у стані об'єкту. Розроблення бази даних – 2019 – С. 30-40.

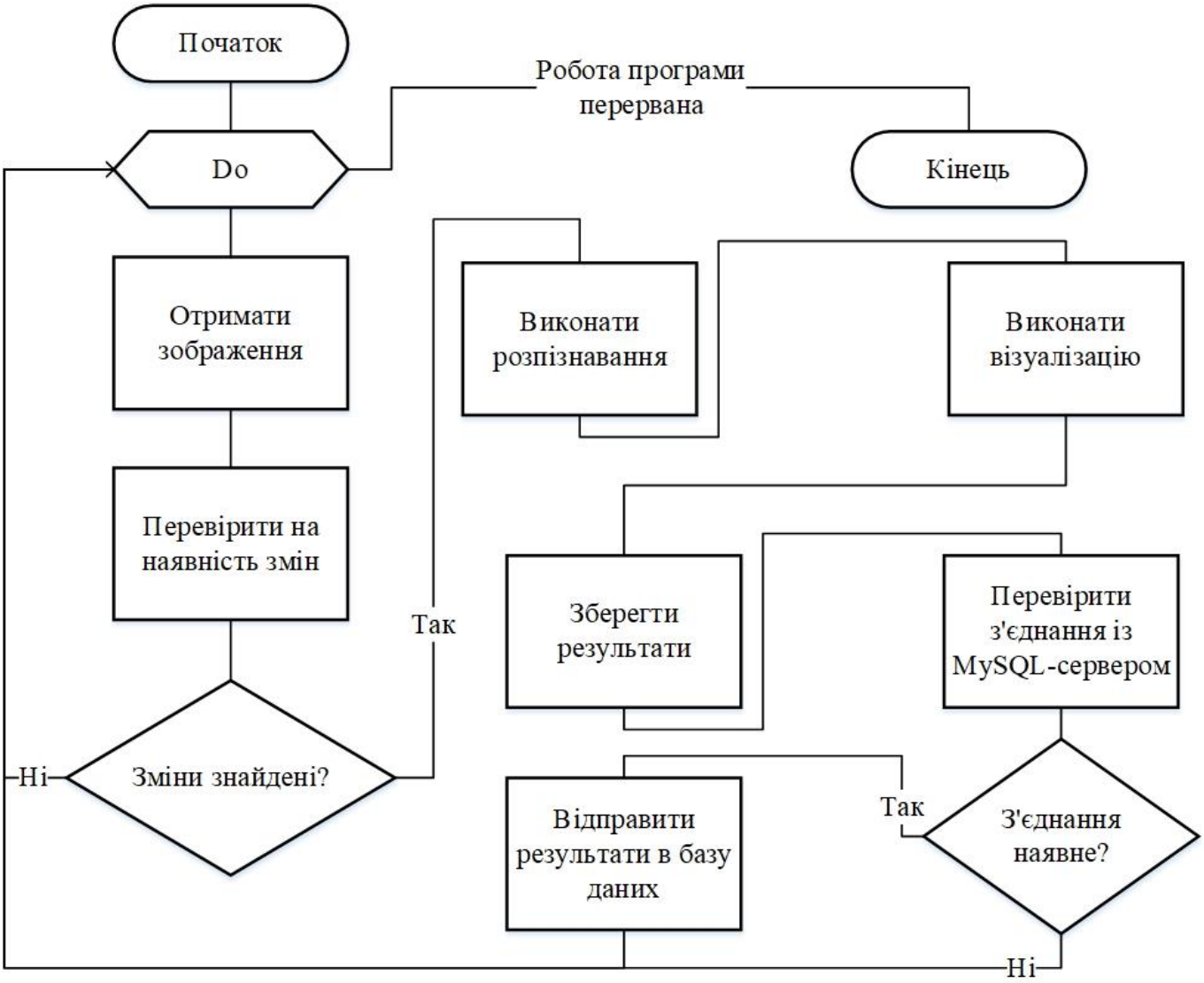
20. І. Кутрес. Hot Chips: Google TPU Performance Analysis Live Blog (3pm PT, 10pm UTC) – 2017 – Отримано з https://www.anandtech.com/show/11749/hot-chips-google-tpu-performance-analysis-live-blog-3pm-pt-10pm-utc.
21. Н. Жоппі. In-Datacenter Performance Analysis of a Tensor Processing Unit – 2018 – New-York, Ithaca, США – 44 Міжнародний Симпозіум з теми: "Архітектура комп'ютера".
22. Дж. Браунлі. What is the Difference Between Test and Validation Datasets? – 2017 – Отримано з https://machinelearningmastery.com/difference-test-validation-datasets/.
23. Г. Джеймс. An Introduction to Statistical Learning with Applications in R / Г. Джеймс, Д. Віттен, Тр. Хасті, Р. Тібширані – 2009 – 176 с. – Springer.
24. Tzutalin. LabelImg – Отримано з https://github.com/tzutalin/labelImg.
25. Mei Сонг. A Mean Field View of the Landscape of Two-Layer Neural Networks / Mei Сонг, А. Монтанарі, Фан-Мін Ньюн – 2018 – №115 (33) – PNAS 10.1073/pnas.1806579115.
26. Pkulzc. Configuring the Object Detection Training Pipeline. Отримано з https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/configuring_jobs.md.
27. Д. Щлегель, Ст. Шапіро. Inference Graphs: A Roadmap – 2013 – США: Департамент Комп'ютерних Наук та Інженерії Університету Баффало, Баффало, Нью-Йорк, 2013 – Друга щорічна конференція на тему: "Досягнення в Когнітивних Системах", Cognitive Systems Foundation.

28. Market Research Vision. Global Gun Rack Market Report, History and Forecast 2014-2025, Breakdown Data by Manufacturers, Key Regions, Types and Application. Отримано з https://www.marketresearchvision.com/reports/171854/Gun-Rack-Market.
29. IBM. A robust AI-centric indoor positioning system – 2018 – Отримано з https://www.ibm.com/blogs/research/2018/10/indoor-positioning-system.
30. Siemens Nixdorf Informationssysteme. Managing devices and Reliant UNIX 5.44. System Administration and Hardware Configuration Using the SYSADM User – 1998 – C. 33-82.

ДОДАТКИ

Додаток 1

Копії графічних матеріалів

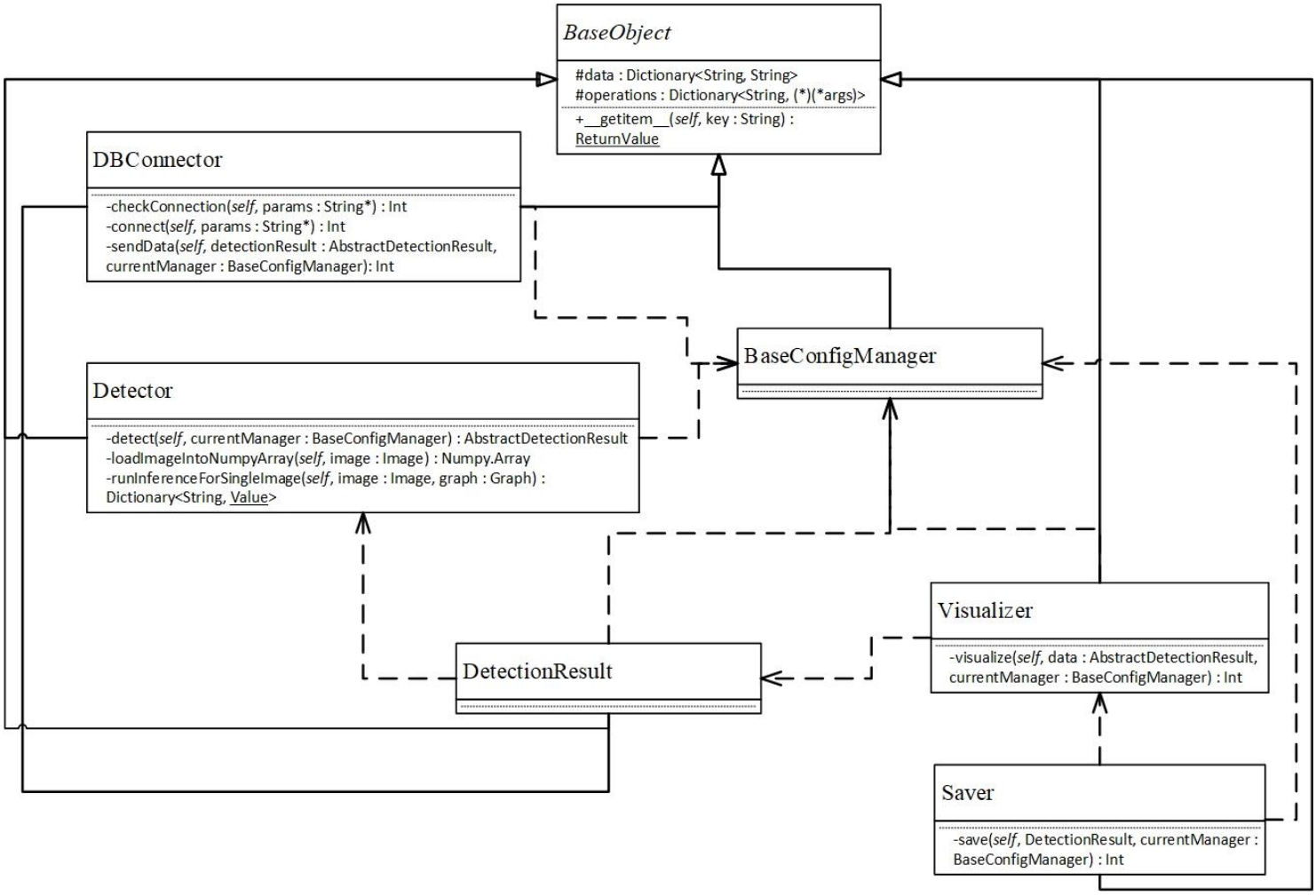


ДП.045200-06-99

Автоматизована система динамічного визначення змін у стані об’єкта. Модуль аналізу та оптимізації.

Схема роботи алгоритму.

Схема алгоритму

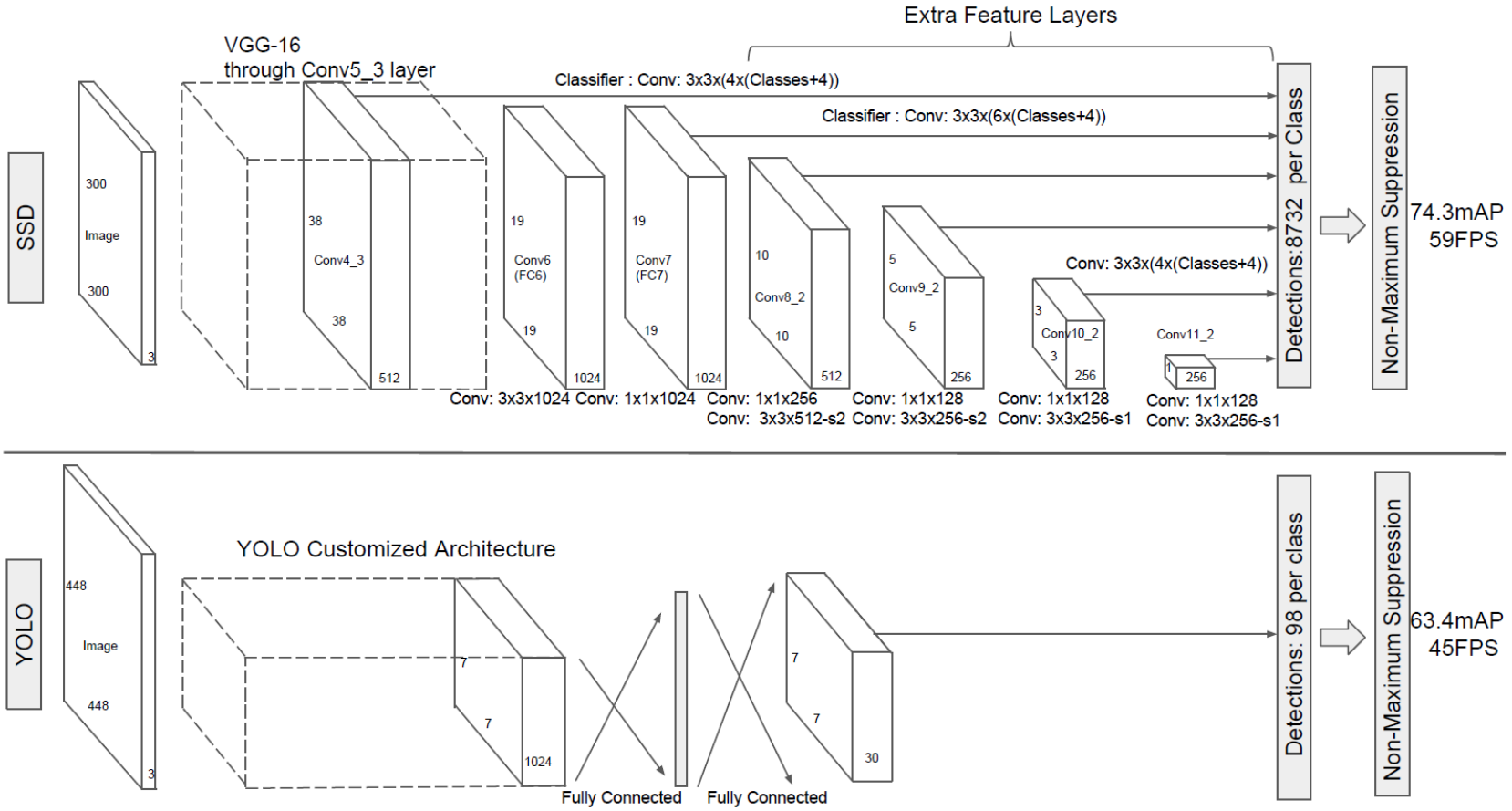


ДП.045200-07-99

Автоматизована система динамічного визначення змін у стані об’єкта. Модуль аналізу та оптимізації.

Діаграма класів модулю аналізу.

Діаграма класів



ДП.045200-08-99.
Автоматизована система динамічного визначення змін у стані об’єкта.
Модуль аналізу та оптимізації.
Схема моделей YOLO та SSD.
Схема алгоритму

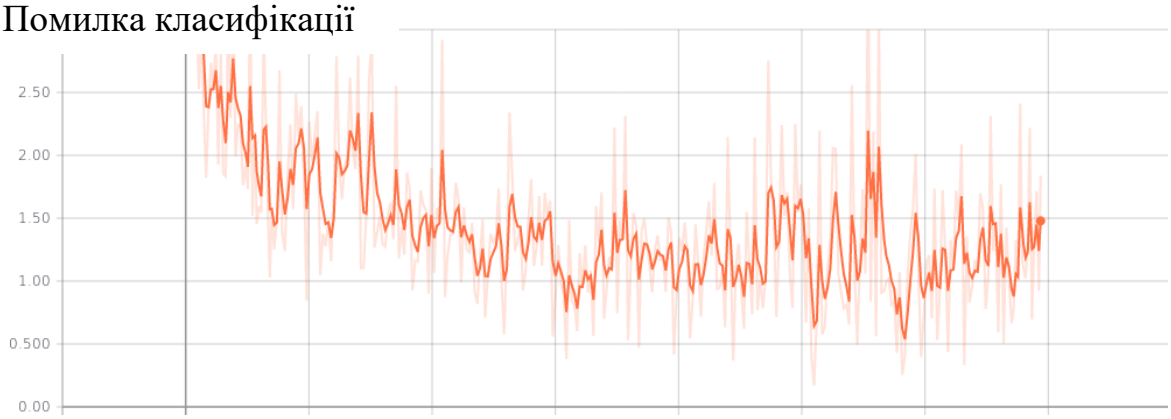
Результати розпізнавання

Вихідні дані

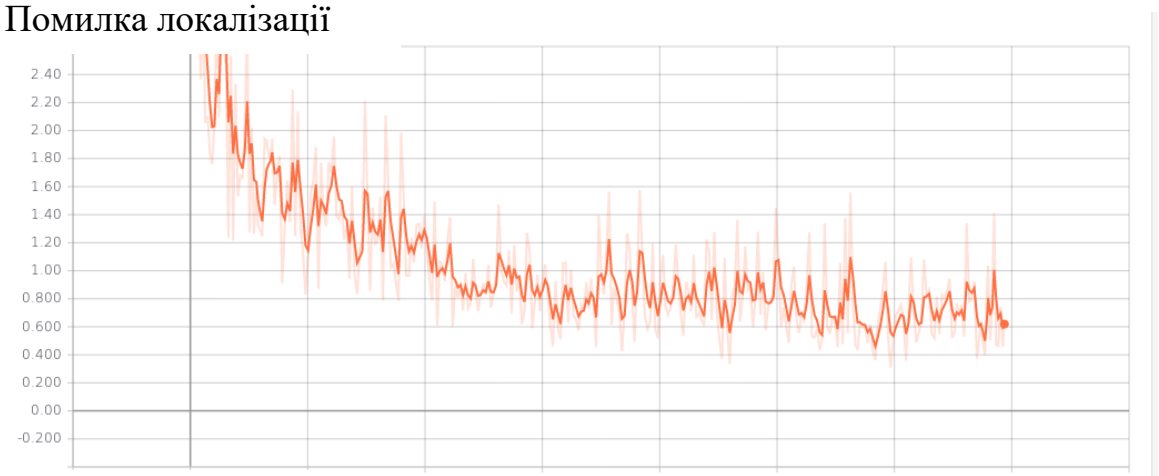
Результати розпізнавання

Іващенко М.В., група КП-51

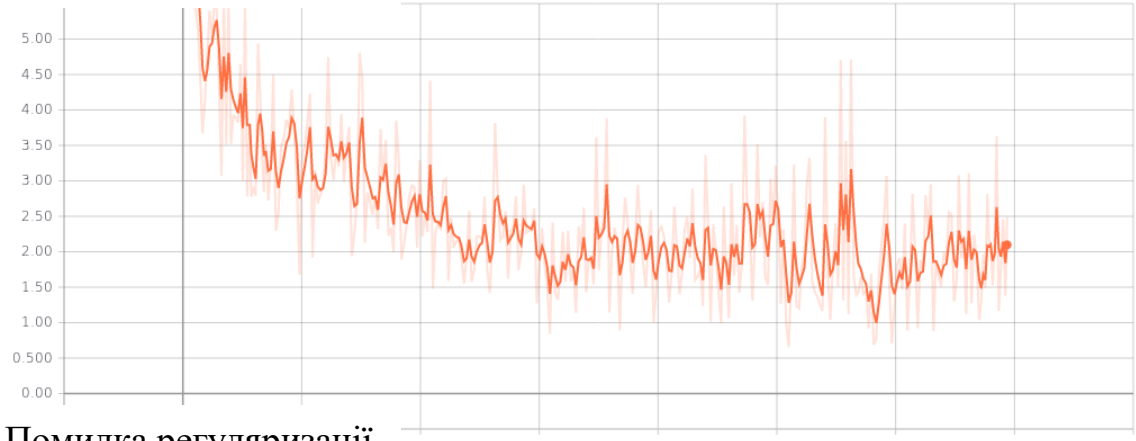
Помилка класифікації



Помилка локалізації



Помилка співпадінь



Помилка регуляризації

Додаток 2

Текст програми

2.1. Код змін, внесених у метод dbse_global_init модулю motion.c.

```
void establishConnection(){
      MYSQL* connection;

      connection = mysql_init(NULL);

      if (!mysql_real_connect(connection, GET_IP, GET_DBUSER0,
            GET_USER_PASSWORD0, GET_DBNAME0, 0, NULL, 0))
      {
            fprintf(stderr, "Connection was aborted. %s\n",
                        mysql_error(connection));
            exit(1);
      }
      printf("Connection established\n");

      mysql_close(connection);
}

static void dbse_global_init(void)
{
…
#ifdef HAVE_MYSQL
    if (mysql_library_init(0, NULL, NULL)) {
        fprintf(stderr, "could not initialize MySQL library");
        exit(1);
    }

    establishConnection();
…
}
```

2.2. Частина функції, яка реалізовує в собі алгоритм обчислення різниці між зображеннями.

```
int alg_diff_standard(struct context *cnt, unsigned char *new)
{
    struct images *imgs = &cnt->imgs;
    int i, diffs = 0;
    int noise = cnt->noise;
    int smartmask_speed = cnt->smartmask_speed;
    unsigned char *ref = imgs->ref;
    unsigned char *out = imgs->img_motion.image_norm;
    unsigned char *mask = imgs->mask;
    unsigned char *smartmask_final = imgs->smartmask_final;
    int *smartmask_buffer = imgs->smartmask_buffer;
```

```
#ifdef HAVE_MMX
    mmx_t mmtemp;    int unload;
#endif

    i = imgs->motionsize;
    memset(out + i, 128, i / 2);

    memset(out, 0, i);

#ifdef HAVE_MMX

    mmtemp.uw[0] = mmtemp.uw[1] = mmtemp.uw[2] = mmtemp.uw[3] =
                   (unsigned short)(noise * 255 + 254);

    movq_m2r(mmtemp, mm7);              /* U */
    pcmpeqb_r2r(mm6, mm6);              /* V */

    pxor_r2r(mm5, mm5);                 /* U */
    psrlw_i2r(8, mm6);                  /* V */

    unload = 255;

    for (; i > 7; i -= 8) {
        movq_m2r(*ref, mm0);
        pxor_r2r(mm4, mm4);
        movq_m2r(*new, mm1);
        movq_r2r(mm0, mm2);

        psubusb_r2r(mm1, mm0);
        psubusb_r2r(mm2, mm1);

        /* Each byte dX in mm0 is abs(nX-rX). */
        por_r2r(mm1, mm0);

        movq_r2r(mm0, mm1);
        punpcklbw_r2r(mm4, mm0);

        punpckhbw_r2r(mm4, mm1);
        if (mask) {

            movq_m2r(*mask, mm2);
            movq_r2r(mm2, mm3);
            punpcklbw_r2r(mm4, mm2);

            punpckhbw_r2r(mm4, mm3);
            pmullw_r2r(mm2, mm0);
            pmullw_r2r(mm3, mm1);

            mask += 8;
        } else {
            movq_r2r(mm0, mm2);
```

```
            psllw_i2r(8, mm0);

            movq_r2r(mm1, mm3);
            psllw_i2r(8, mm1);

            psubusw_r2r(mm2, mm0);      /* U */
            psubusw_r2r(mm3, mm1);      /* V */
        }

        psubusw_r2r(mm7, mm0);
        psubusw_r2r(mm7, mm1);          /* V */

        pcmpeqw_r2r(mm4, mm0);
        pcmpeqw_r2r(mm4, mm1);          /* V */

        pand_r2r(mm6, mm0);
        pand_r2r(mm6, mm1);             /* V */

        pxor_r2r(mm6, mm0);             /* U: invert the result */
        pxor_r2r(mm6, mm1);             /* V */

        packuswb_r2r(mm1, mm0

       /* SMARTMASK PROCESSING */

        movq_m2r(*new, mm2);
        pand_r2r(mm0, mm2);              /* U: mm1 = N7 N6 N5 N4 N3 N2
N1 N0 */
        psubb_r2r(mm0, mm4);            /* V: mm4 = 0x01 where dX>noise
*/

        movq_r2m(mm2, *out);             /* U: this will stall */
        paddusb_r2r(mm4, mm5);          /* V: add counts to mm5 */

        /*
         * Every 255th turn, we need to unload mm5 into the diffs
variable,
         * because otherwise the packed bytes will get saturated.
         */
        if (--unload == 0) {
            /* Unload mm5 to memory and reset it. */
            movq_r2m(mm5, mmtemp);      /* U */
            pxor_r2r(mm5, mm5);         /* V: mm5 = 0 */

            diffs += mmtemp.ub[0] + mmtemp.ub[1] + mmtemp.ub[2] +
            mmtemp.ub[3] + mmtemp.ub[4] + mmtemp.ub[5] + mmtemp.ub[6]
            + mmtemp.ub[7];
            unload = 255;
        }

        out += 8;
```

```
        ref += 8;
        new += 8;
    }

    /*
     * Check if there are diffs left in mm5 that need to be copied to
the
     * diffs variable.
     */
    if (unload < 255) {
        movq_r2m(mm5, mmtemp);
        diffs  +=  mmtemp.ub[0]  +  mmtemp.ub[1]  +  mmtemp.ub[2]  +
mmtemp.ub[3] +
                  mmtemp.ub[4]  +  mmtemp.ub[5]  +  mmtemp.ub[6]  +
mmtemp.ub[7];
    }

    emms();

#endif
    /* NON-MMX CODE */

        out++;
        ref++;
        new++;
    }
    return diffs;
}
```

2.3. Частина функції, яка реалізовує в собі алгоритм усунення шуму.

```
void alg_noise_tune(struct context *cnt, unsigned char *new)
{
    struct images *imgs = &cnt->imgs;
    int i;
    unsigned char *ref = imgs->ref;
    int diff, sum = 0, count = 0;
    unsigned char *mask = imgs->mask;
    unsigned char *smartmask = imgs->smartmask_final;

    i = imgs->motionsize;

    for (; i > 0; i--) {
        diff = ABS(*ref - *new);

        if (mask)
            diff = ((diff * *mask++) / 255);

        if (*smartmask) {
```

```
            sum += diff + 1;
            count++;
        }
        ref++;
        new++;
        smartmask++;
    }

    if (count > 3)
        sum /= count / 3;

    cnt->noise = 4 + (cnt->noise + sum) / 2;
}
```

2.3. Макроси для здійснення маніпуляцій із mmx-регістрами.

```
…

#define    mmx_r2r(op,regs,regd) \
    __asm__ __volatile__ (#op " %" #regs ", %" #regd)

#define    emms() __asm__ __volatile__ ("emms")

#define    movd_m2r(var,reg)     mmx_m2r (movd, var, reg)
#define    movd_r2m(reg,var)     mmx_r2m (movd, reg, var)
#define    movd_r2r(regs,regd)   mmx_r2r (movd, regs, regd)
…
```

2.4. Формат скрипту формування даних для навчання.

```
python3.6 create_pascal_record.py \
        --data_dir=/path_to_data \
        --output_path=/output_path \
        --label_map_path=/map_path
```

2.5. Скелет конфігураційного файлу для навчання нейронної мережі.

```
model {
(/*КОНФІГУРАЦІЯ МОДЕЛІ*/...)
}
train_config : {
(/*КОНФІГУРАЦІЯ МОДЕЛІ*/...)
}
train_input_reader: {
(/*ТРЕНУВАЛЬНА_КОНФІГУРАЦІЯ*/...)
}
eval_config: {
```

```
(/*ПЕРЕВІРОЧНА КОНФІГУРАЦІЯ*/)
}
eval_input_reader: {
()
}
```

2.6. Файл розмітки.

```
item {
  id: 1
  name: "hit"
}
```

2.7. Формат запуску скрипту навчання найронної мережі.

```
./train \
        --logtostderr \
        --train_dir=path_to_train_dir \
        --pipeline_config_path=pipeline_config.pbtxt
```

2.8. Формат запуску скрипту експорту графу виводу.

```
INPUT_TYPE=image_tensor
PIPELINE_CONFIG_PATH={шлях до пайплайн-конфігурації}
TRAINED_CKPT_PREFIX={шлях до model.ckpt}
EXPORT_DIR={шлях до папки, що буде використана для збереження
результатів експорту}
python object_detection/result_inference_graph.py \
    --input_type=${ВХІДНИЙ_ТИП} \
    --pipeline_config_path=${ШЛЯХ_ПАЙПЛАЙН_КОНФІГУРАЦІЇ} \
    --trained_checkpoint_prefix=${ЧЕКПОІНТ_МОДЕЛІ} \
    --output_directory=${КАТАЛОГ_ВИВЕДЕННЯ_ДАНИХ}
```

2.9. Точка входу програми, що реалізовує в собі модуль аналізу.

```
# GeneralConfig file.
# Contains startup configuration parameters and imports
from GeneralConfig import *

# BaseRMManager object. Defines data allocation in memory
generalRMManager = BaseRMManager()

# ShotDetector object. Performs the detection process
shotDetector = ShotDetector()

# BaseDBConnector object. Communicates with the database
dbConnector = DBConnector()
```

```
# BaseVizualizer object. Visualizes the detected data upon the image
visualizer = DetectionDataVisualizer()

# Saves the data according to the selected format
saver = DetectionDataSaver()

# Commit detection
detectionResult = shotDetector[COMMIT_DETECTION]
                    (generalRMManager[PATH_PARAMS])
# Send results to the database
dbConnector[SEND_TO_DATABASE](detectionResult[SHOT_DATA])
# Save the results locally
saver[SAVE_DATA](visualizer[VISUALIZE_DATA](detectionResult))
```

2.10. Завантаження зображення в numpy-array.

```
def load_image_into_numpy_array(self, image):
        (im_width, im_height) = image.size
        return np.array(image.getdata()).reshape(
            (im_height, im_width, 3)).astype(np.uint8)
```

2.11. Алгоритм розбору моделі нейронної мережі.

```
def run_inference_for_single_image(self, image, graph):
        with graph.as_default():
            with tf.Session() as sess:
                # Get handles to input and output tensors
                ops = tf.get_default_graph().get_operations()
                all_tensor_names = {output.name for op in ops for
                output in op.outputs}
                tensor_dict = {}
                for key in [
                      'num_detections', 'detection_boxes',
                      'detection_scores',
                      'detection_classes', 'detection_masks'
                    ]:
                    tensor_name = key + ':0'
                    if tensor_name in all_tensor_names:
                      tensor_dict[key] =
                      tf.get_default_graph().get_tensor_by_name(
                            tensor_name)
                if 'detection_masks' in tensor_dict:
                     detection_boxes =
                     tf.squeeze(tensor_dict['detection_boxes'], [0])
                    detection_masks =
                    tf.squeeze(tensor_dict['detection_masks'], [0])

                    real_num_detection =
                    tf.cast(tensor_dict['num_detections'][0],
                    tf.int32)
```

```
                    detection_boxes = tf.slice(detection_boxes,
                    [0, 0], [real_num_detection, -1])
                    detection_masks = tf.slice(detection_masks,
                    [0, 0, 0], [real_num_detection, -1, -1])
                    detection_masks_reframed =
                    utils_ops.reframe_box_masks_to_image_masks(
                        detection_masks, detection_boxes,
                        image.shape[1], image.shape[2])
                    detection_masks_reframed = tf.cast(
                      tf.greater(detection_masks_reframed, 0.5),
                      tf.uint8)

                    tensor_dict['detection_masks'] = tf.expand_dims(
                        detection_masks_reframed, 0)
                    image_tensor =
                    tf.get_default_graph().get_tensor_by_name('image
                    _tensor:0')

                # Run inference
                output_dict = sess.run(tensor_dict,
                    feed_dict={image_tensor: image})


                output_dict['num_detections'] =
                int(output_dict['num_detections'][0])
                output_dict['detection_classes'] = output_dict[
                    'detection_classes'][0].astype(np.uint8)
                output_dict['detection_boxes'] =
                      output_dict['detection_boxes'][0]
                output_dict['detection_scores'] =
                      output_dict['detection_scores'][0]
                if 'detection_masks' in output_dict:
                      output_dict['detection_masks'] =
                      output_dict['detection_masks'][0]
                return output_dict
```

2.12. Алгоритм розпізнавання.

```
def detect(self, pathParams):
        detectionResult = ShotDetectionResult()

        detection_graph = tf.Graph()
        with detection_graph.as_default():
            od_graph_def = tf.GraphDef()
            with tf.gfile.GFile(pathParams[PATH_TO_CKPT], 'rb') as
                 fid:
                serialized_graph = fid.read()
                od_graph_def.ParseFromString(serialized_graph)
                tf.import_graph_def(od_graph_def, name='')
```

```
            label_map =
            label_map_util.load_labelmap(pathParams[PATH_TO_LABE
            LS])
            categories =
            label_map_util.convert_label_map_to_categories(label
            _map, max_num_classes=pathParams[NUM_CLASSES],
            use_display_name=True)
            category_index =
            label_map_util.create_category_index(categories)

    with detection_graph.as_default():
        with tf.Session(graph=detection_graph) as sess:
            image_tensor =
            detection_graph.get_tensor_by_name('image_tensor:0')

            count = 1
            for image_path in pathParams[IMAGE_PATH]:
               image = Image.open(image_path)
                    image_width, image_height = image.size

               image_np = load_image_into_numpy_array(image)

               image_np_expanded = np.expand_dims(image_np,
               axis=0)

                # Actual detection.
               output_dict =
               run_inference_for_single_image(image_np_expanded,
               detection_graph)

               detectionResult[BOXES] =
               output_dict['detection_boxes']
               detectionResult[CLASSES] =
               output_dict['detection_classes']
               detectionResult[SCORES] =
               output_dict['detection_scores']

    return detectionResult
```

Додаток 3

Копія презентації

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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

АВТОМАТИЗОВАНА СИСТЕМА ДИНАМІЧНОГО ВИЗНАЧЕННЯ ЗМІН У СТАНІ ОБ’ЄКТА. МОДУЛЬ АНАЛІЗУ ТА ОПТИМІЗАЦІЇ

Виконав: Іващенко Михайло Вікторович

Науковий керівник: ст. викладач, к.т.н., Люшенко Леся Анатоліївна

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмну систему, що розгортається на апаратному пристрої, яка б автоматично визначала та позиціонувала зміни в стані об’єкту. В якості досліджуваного об’єкту обрано стрілецьку мішень, в якості змін, що необхідно визначити – влучення

ПОСТАНОВКА ЗАДАЧІ

Завдання:

1. Розробити модуль аналізу та оптимізації системи:
   1.1. На основі встановлених вимог
   1.2. В рамках використання концептуальної моделі
2. Проаналізувати існуючі рішення
3. Зв’язати модуль з іншими частинами системи
4. Провести як незалежне тестування модулю, так і тестування в рамках системи

АКТУАЛЬНІСТЬ

Сфера впровадження розробки: ручний процес збору та аналізу даних з мішені вимагає суттєвих витрат ресурсів та часу

Сфери впровадження внутрішньої технології:

- Медицина
- Астрономія
- Охоронні системи
- Позиціонування об’єктів
- Системи моніторингу

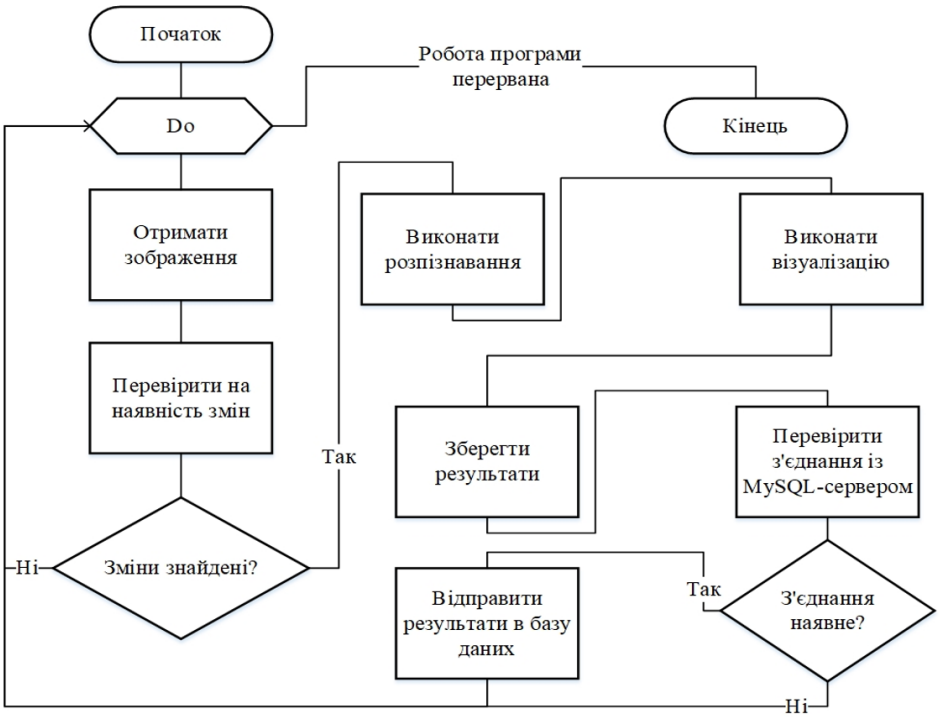
АРХІТЕКТУРА ЯДРА

Модель ядра: «Предиктор-Коректор»

Внутрішні модулі:

1. Модуль контролю:
   - Здійснює запуск модулю
   - Отримує дані з відеопотоків
   - Втілює функції «Коректора»
2. Модуль аналізу:
   - Здійснює розпізнавання
   - Зберігає результати
   - Комунікує із сервером баз даних

ЗАГАЛЬНИЙ АЛГОРИТМ РОБОТИ

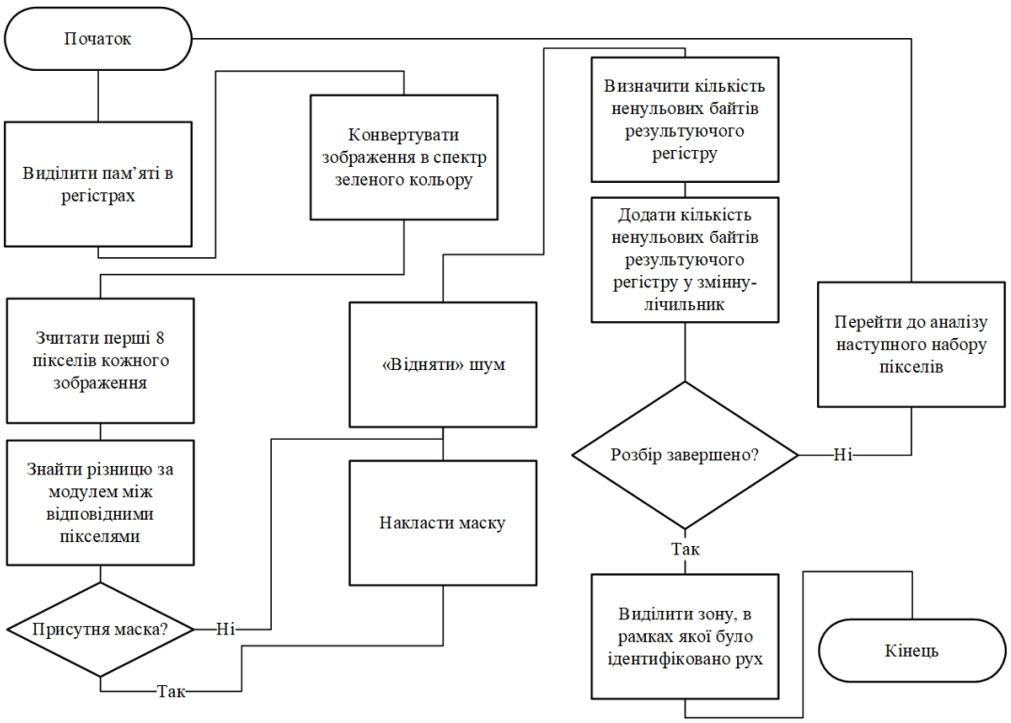


ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

МОДУЛЬ УПРАВЛІННЯ

- Розроблено на основі ПЗ з відкритим кодом
- Motion розповсюджується за GPU-ліцензією
- Засоби розробки: C/C++
- Причини використання:
  - Сформовані алгоритми взаємодії із апаратною платформою
  - Сформовані алгоритми управління зовнішніми пристроями
  - Алгоритми реалізовані засобами низькорівневого програмування

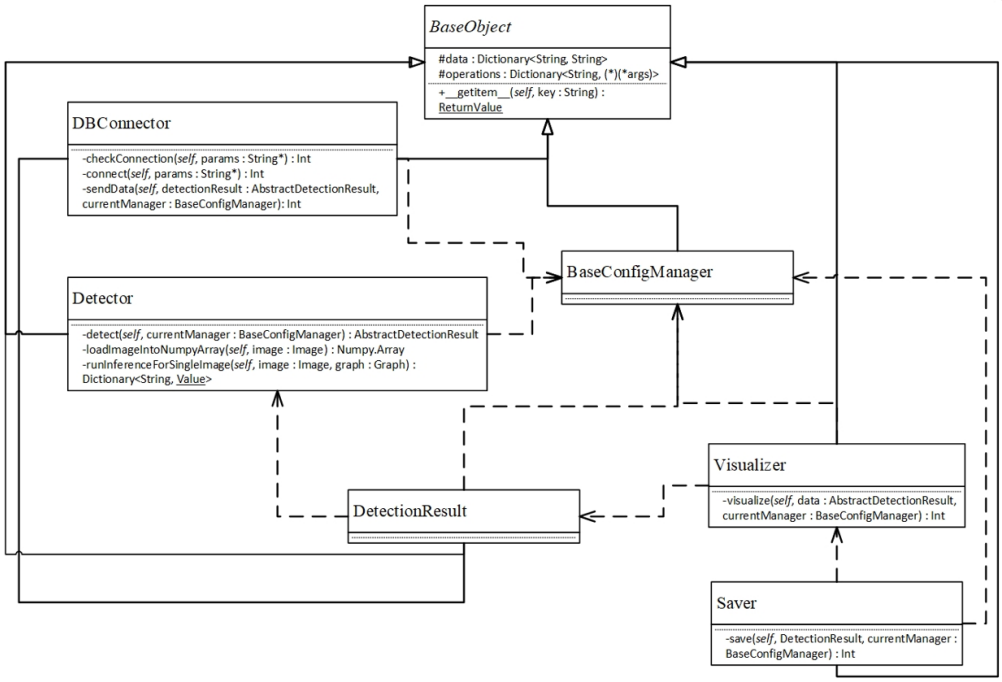
АЛГОРИТМ ГРАФІЧНОЇ ОБРОБКИ



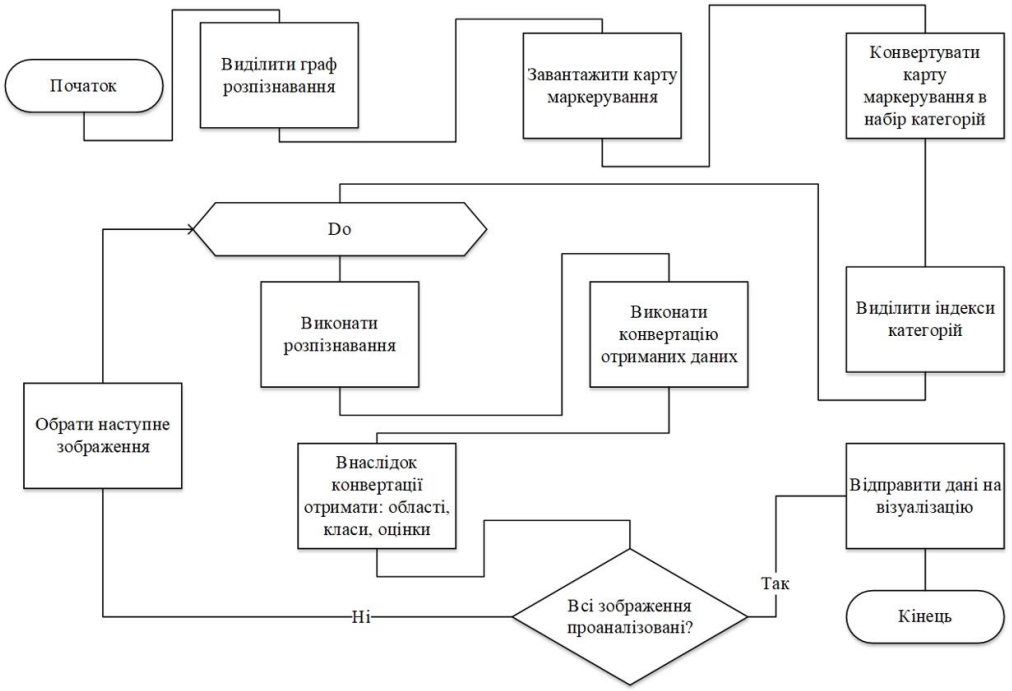
МОДУЛЬ АНАЛІЗУ

- Розроблено на основі бібліотек Tensorflow та Tensorflow Object Detection API
- Засоби розробки: python версії 3.5.0
- Причини використання:
  - Ефективний інструментарій роботи з нейронними мережами
  - Зручні інтерфейси для візуалізації даних
  - Можливість побудови об'єктно-орієнтованої архітектури

ДІАГРАМА КЛАСІВ



АЛГОРИТМ РОЗПІЗНАВАННЯ



ГІПЕРПАРАМЕТРИ МЕРЕЖІ

- Кількість класів: 1
- Виключення: відсутнє
- Функція активації: RELU_6
- Мінімальна глибина: 16
- Ініціалізація вагів: рівномірний розподіл
- Темп навчання: 0.004
- Імпульс: 0.9
- Розмір кадру: 4

АПАРАТНА ЧАСТИНА

Мікрокомп’ютер: Raspberry Pi – Raspberry Pi 3 Model B

Основні функції:

- Є місцем розгортання ПЗ, розроблюваного в рамках модулю
- Точка підключення камер
- Здійснює апаратне підключення до інших пристроїв системи

Камери: Logitech Webcam C500

Параметри:

- 1.3 Mp
- RightSound, RightLight Technologies

КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО МОДУЛЮ

1. Основні критерії:

1.1. Відповідність функціональним вимогам

1.2. Стабільність роботи

1.3. Можливість модифікації

1.4. Швидкодія

1.5. Точність розпізнавання

1.6. Портативність

2. Вторинні критерії:

2.1. Документованість

2.2. Важкість розгортання

КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО МОДУЛЮ

1.1. Виконується: визначення та позиціонування влучень, відправлення даних у БД для подальшого виведення в інтерфейсі користувача

1.2. Мікроконтролер працює при повному навантаженні до 10 годин без залучення системи охолодження

1.3. Код та загальна модель модулю спроектовані за концептуальними принципами, що полегшує модифікацію

1.4. Один цикл роботи системи складає 10 секунд

КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО МОДУЛЮ

1.5. Середня точність розпізнавання складає 75% (визначених влучень з усіх наявних в мішені)

1.6. Модуль має високий рівень портативність за рахунок простої апаратної структури: мікрокомп’ютер, камера

2.1. Система добре документується за рахунок наявності засобів створення автоматичної документації до коду

2.2. Модуль легко розгортається: апаратний комплект, з’єднувальні Ethernet-кабелі

17

ПОРІВНЯННЯ З АНАЛОГАМИ ЗА КРІТЕРІЯМИ

	Розроблений модуль	SCATT	Лазерне розпізнавання
Функціональність	**9**	6	10
Стабільність	**9**	9	10
Модифікованість	**10**	3	6
Швидкодія	**7**	6	10
Точність	**7**	7	10
Портативність	**10**	7	3
Документованість	**10**	5	2
Процес розгортання	**10**	6	3

Оціночна шкала:

18

ВИСНОВКИ

1. Розроблено модуль аналізу та оптимізації системи динамічного визначення влучень у мішень
2. Спроектовано архітектуру модель модулю із врахуванням сформованих вимог
3. Розроблено та розгорнуто алгоритми аналізу із залученням комп'ютерного зору та штучного інтелекту
4. Протестовано та порівняно з аналогами за критеріями: функціональними, стабільності, модифікованості тощо
5. Визначено шляхи вдосконалення системи та технології

ДОСЯГНЕННЯ ТА ПУБЛІКАЦІЇ

Нагороди:

1. Фіналіст та переможець фестивалю інноваційних розробок «Sikorsky Challenge 2018»
2. Фіналіст фестивалю воєнних інноваційних розробок «Sikorsky Challenge 2019. Military»
3. Підписано маніфест про наміри з фондом Міхалевича
4. Лауреат премії президентського фонду «Україна» Леоніда Кучми

Публікації:

«Integer Norm for Difference Assessment of the Frame Elements Considering the White Balance», подана в редакцію наукового журналу «Управляющие системы и машины»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

АВТОМАТИЗОВАНА СИСТЕМА ДИНАМІЧНОГО ВИЗНАЧЕННЯ ЗМІН У СТАНІ ОБ'ЄКТА. МОДУЛЬ АНАЛІЗУ ТА ОПТИМІЗАЦІЇ

Програма та методика тестування

ДП.045200-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.В. Іващенко

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Автоматизована система динамічного визначення змін у стані об'єкта. Модуль аналізу та оптимізації, який являє собою ядро, створене в рамках системи з метою виконання динамічного позиціонування влучень у мішень під час стрільб.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- апаратна компактність;
- дистанційна передача даних;
- коректність роботи бібліотек;
- результати розпізнавання влучень;
- визначення послідовності влучень;
- взаємодія ядра з іншими модулями системи.

3. МЕТОДИ ТЕСТУВАННЯ

Виконуються модульне, інтеграційне та системне тестування.

1. Мета модульного тестування: перевірка вірної роботи бібліотек, на основі яких здійснюється розробка, а також модулів, що реалізують в собі частини моделі «Предиктор» та «Коректор»;
2. Мета інтеграційного тестування: перевірити коректність роботи моделі «Предиктор-Коректор в цілому», перевірити основні функції системи, що стосуються розпізнавання;
3. Мета системного тестування: перевірити роботу ядра в рамках всієї системи шляхом з’єднання відповідного пристрою із сервером баз даних та веб-сервером; необхідно при цьому перевірити

коректність змін даних всередині бази даних та відповідну зміну інтерфейсу користувача.

Використовуються наступні методи:

- тестування установки;
- функціональне тестування;
- тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності);
- конфігураційне тестування;
- тестування взаємодії.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально. Працездатність ядра системи перевіряється наступним чином:

1. Тестування коректної роботи бібліотек шляхом перевірки правильної роботи набору тестових прикладів.
2. Тестування коректної роботи підключених пристроїв шляхом перевірки їх ідентифікації всередині операційної системи.
3. Тестування коректності розгортання в рамках мікроконтролера шляхом перевірки роботи із різними конфігураційними даними.
4. Статичне тестування коду.
5. Динамічне тестування на відповідність функціональним вимогам.
6. Тестування ядра із різними зовнішніми пристроями.
7. Тестування ядра при взаємодії із іншими частинами системи.
8. Тестування стабільності роботи.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

АВТОМАТИЗОВАНА СИСТЕМА ДИНАМІЧНОГО ВИЗНАЧЕННЯ ЗМІН У СТАНІ ОБ'ЄКТА. МОДУЛЬ АНАЛІЗУ ТА ОПТИМІЗАЦІЇ

Керівництво користувача

ДП.045200-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ М.В. Іващенко

2019

ЗМІСТ

1. ДОСТУП ДО МОЖЛИВОСТЕЙ МОДУЛЮ

Доступ до можливостей модулю аналізу та оптимізації можливо отримати безпосередньо при взаємодії з мікроконтролером (мікрокомп’ютером) за рахунок виконання запуску системи з командного рядку. Результати розпізнавання можливо переглянути як локально, так і всередині веб-інтерфейсу, що формується в рамках комунікативного модулю системи.

2. ОПИС ФУНКЦІЙ ТА МОЖЛИВОСТЕЙ МОДУЛЮ

Всередину модулю закладені наступні функції:

- є точкою підключення камер;
- виконання передграфічної обробки;
- виконання розпізнавання;
- відправлення результатів у модуль комунікації.

Таким чином, модуль реалізовує в собі виконання наступних вимог:

- здійснює запуск ядра систем;
- визначає та позиціонує влучення всередині мішені;
- визначає послідовність влучень;
- відправляє дані на сервер до подальшого виведення в інтерфейсі користувача.

2.1. Запуск модулю

Запуск модулю здійснюється при безпосередній роботі з мікрокомп’ютером. Для цього достатньо здійснити запуск підмодулю Motion з відповідного каталогу (див. рис. 1).

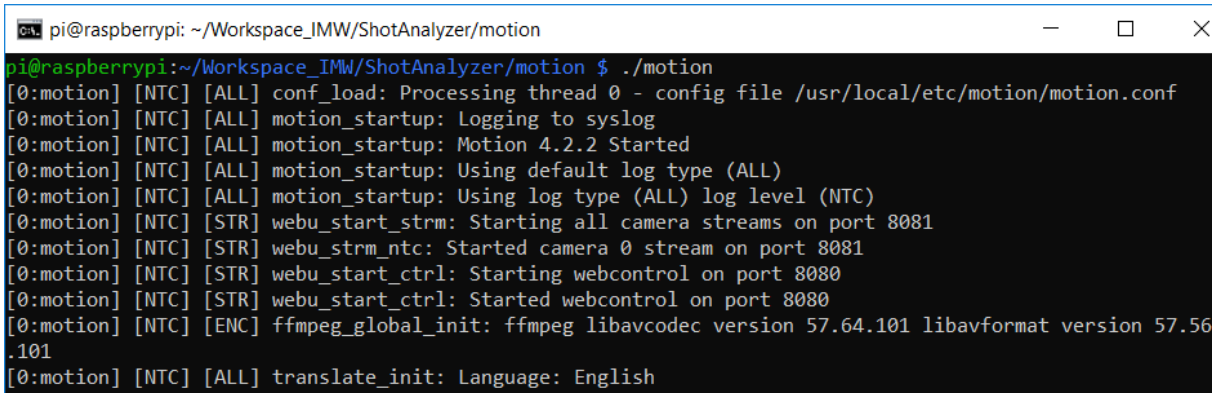


Рис. 1. Запуск ядра системи

2.2. Ініціалізація відеопристроїв

Після запуску, здійснюється ініціалізація пристроїв, які є джерелам відеопотоку. Логування демонструє, що до пристрою підключено камеру, яка вибрана в якості джерела даних (див. рис. 2).

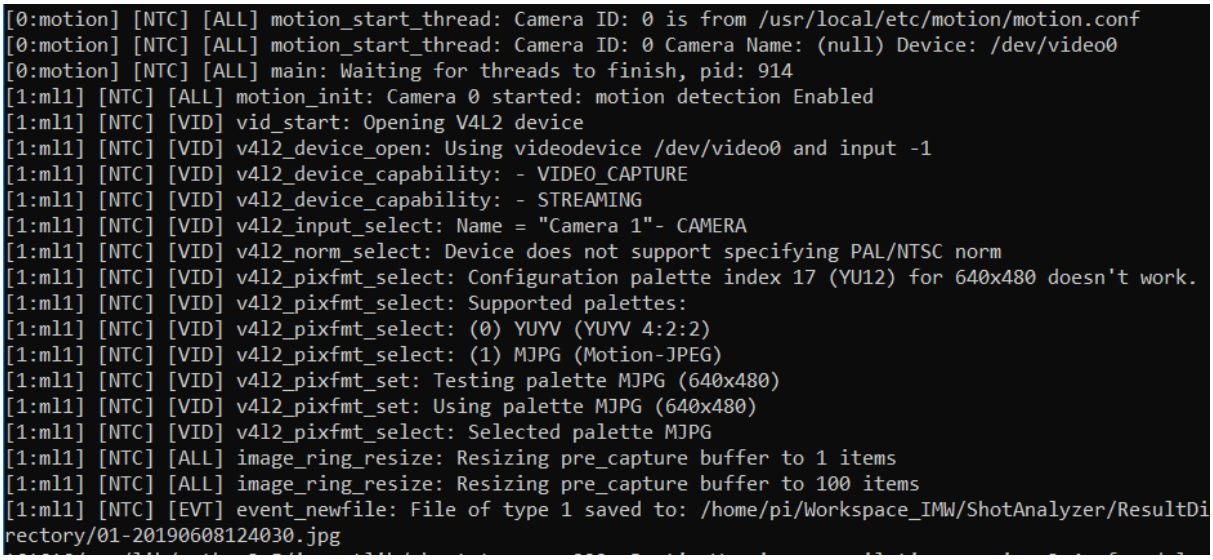


Рис. 2. Ініціалізація відеопристроїв

За необхідності підключення декількох камер, необхідно додати запис у відповідний конфігураційний файл (див. рис. 3).

```
# Target directory for pictures, snapshots and movies
target_dir /home/pi/Workspace_IMW/ShotAnalyzer/ResultDirectory

# Video device (e.g. /dev/video0) to be used for capturing.
videodevice /dev/video0

# Parameters to control video device.  See motion_guide.html
; vid_control_params value
```

Рис. 3. Фіксація відеопристроїв всередині конфігураційного файлу

За необхідності перевірки підключення того чи іншого відеопристрою, достатньо перевірити каталог /dev/ командою “ls dev”, в якому знаходить перелік підключених пристроїв. Відеопристрої, зазвичай, отримують назву “videon”, де *n* – це порядковий номер пристрою в системі (див. рис. 4).

```
pi@raspberrypi:/usr/local/etc/motion $ ls /dev
autofs          kmsg              network_throughput  random   tty19  tty38  tty57      vcs1
block           log               null                raw      tty2   tty39  tty58      vcs2
btrfs-control   loop0             ppp                 rfkill   tty20  tty4   tty59      vcs3
bus             loop1             ptmx                serial1  tty21  tty40  tty6       vcs4
cachefiles      loop2             pts                 shm      tty22  tty41  tty60      vcs5
char            loop3             ram0                snd      tty23  tty42  tty61      vcs6
console         loop4             ram1                stderr   tty24  tty43  tty62      vcs7
cpu_dma_latency loop5             ram10               stdin    tty25  tty44  tty63      vcsa
cuse            loop6             ram11               stdout   tty26  tty45  tty7       vcsa1
disk            loop7             ram12               tty      tty27  tty46  tty8       vcsa2
fb0             loop-control      ram13               tty0     tty28  tty47  tty9       vcsa3
fd              mapper            ram14               tty1     tty29  tty48  ttyAMA0    vcsa4
full            media0            ram15               tty10    tty3   tty49  ttyprintk  vcsa5
fuse            mem               ram2                tty11    tty30  tty5   uhid       vcsa6
gpiochip0       memory_bandwidth  ram3                tty12    tty31  tty50  uinput     vcsa7
gpiochip1       mmcblk0           ram4                tty13    tty32  tty51  urandom    vcsm
gpiochip2       mmcblk0p1         ram5                tty14    tty33  tty52  v4l        vhci
gpiomem         mmcblk0p2         ram6                tty15    tty34  tty53  vchiq      video0
hwrng           mqueue            ram7                tty16    tty35  tty54  vcio       watchdog
initctl         net               ram8                tty17    tty36  tty55  vc-mem     watchdog0
input           network_latency   ram9                tty18    tty37  tty56  vcs        zero
```

Рис. 4. Вміст каталогу dev

2.3. Процес розпізнавання

Протягом роботи модулю, Object Detection API може виконувати оптимізацію тих чи інших параметрів при виконанні розпізнавання (див. рис. 5).

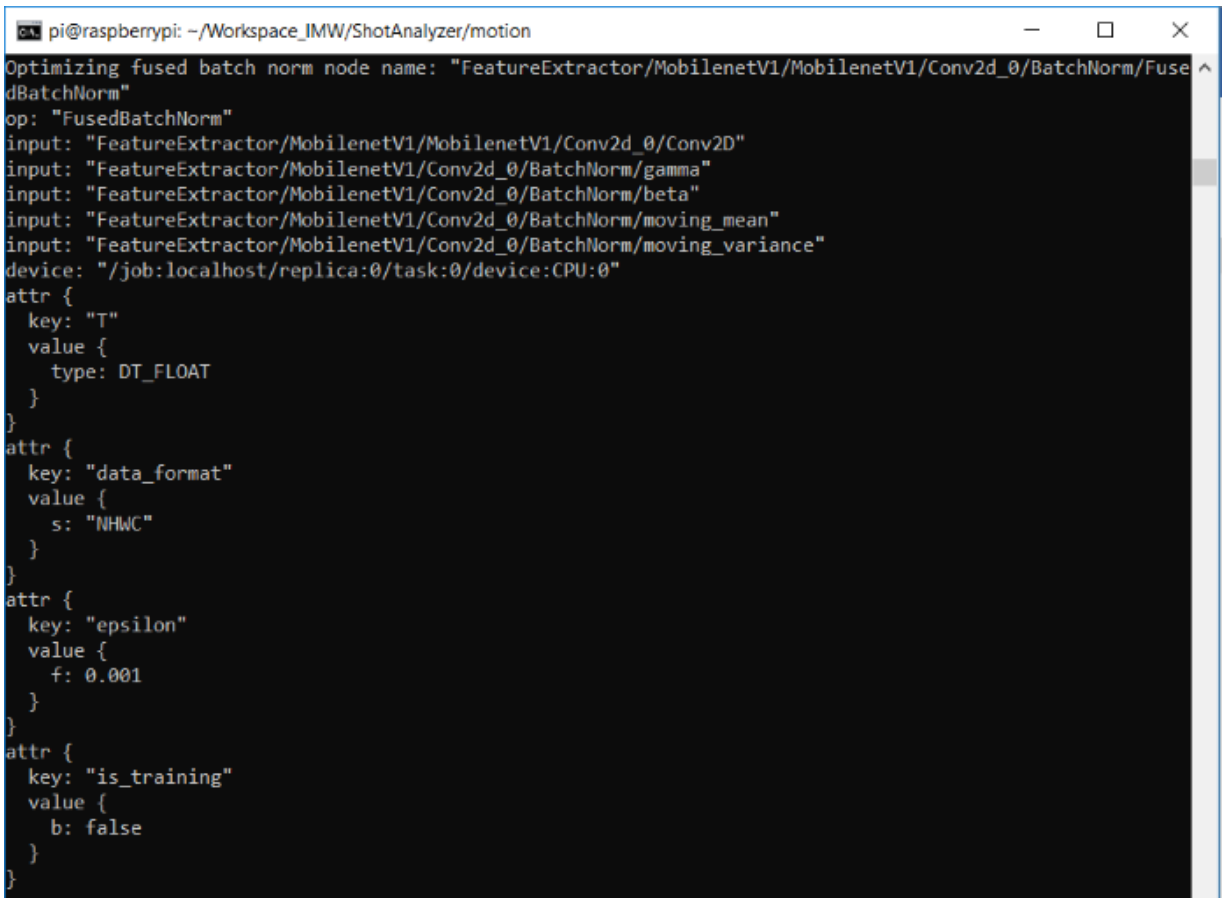


Рис. 5. Повідомлення про виконання оптимізації параметрів

Візуалізовані результати розпізнавання зберігаються в каталог. Назва кожного файлу при цьому відповідає єдиному формату: “№_камери-рік/місяць/день/години/хвилини/секунди.jpg”. Копія результатів останнього проаналізованого зображення також зберігається під іменем “lastsnap.jpg” (доступ до даного файлу надається пристрою, на якому розташовано веб-сервер з метою його подальшого відображення в інтерфейсі користувача) (див рис. 6, 7, 8).

```
pi@raspberrypi:~/Workspace_IMW/ShotAnalyzer/ResultDirectory $ ls
01-20190511024200.jpg  01-20190511043700.jpg  01-20190511064700.jpg  01-20190511071930.jpg
01-20190511024530.jpg  01-20190511043800.jpg  01-20190511064830.jpg  01-20190511072030.jpg
01-20190511024600.jpg  01-20190511043900.jpg  01-20190511064930.jpg  01-20190511072130.jpg
01-20190511025131.jpg  01-20190511044000.jpg  01-20190511065000.jpg  01-20190511072230.jpg
01-20190511025330.jpg  01-20190511044100.jpg  01-20190511065100.jpg  01-20190511072330.jpg
01-20190511025430.jpg  01-20190511044200.jpg  01-20190511065200.jpg  01-20190511072430.jpg
01-20190511025530.jpg  01-20190511044300.jpg  01-20190511065300.jpg  01-20190511072700.jpg
01-20190511034331.jpg  01-20190511044400.jpg  01-20190511065400.jpg  01-20190511072800.jpg
01-20190511034501.jpg  01-20190511044500.jpg  01-20190511065500.jpg  01-20190511072900.jpg
01-20190511040830.jpg  01-20190511044600.jpg  01-20190511065600.jpg  01-20190511110600.jpg
01-20190511041001.jpg  01-20190511044700.jpg  01-20190511065700.jpg  01-20190511110700.jpg
01-20190511041100.jpg  01-20190511044800.jpg  01-20190511065800.jpg  01-20190511110800.jpg
01-20190511041200.jpg  01-20190511044900.jpg  01-20190511065900.jpg  01-20190511110900.jpg
01-20190511041301.jpg  01-20190511045000.jpg  01-20190511070000.jpg  01-20190511111000.jpg
01-20190511041400.jpg  01-20190511045100.jpg  01-20190511070100.jpg  01-20190511111100.jpg
01-20190511041500.jpg  01-20190511045200.jpg  01-20190511070200.jpg  01-20190511111200.jpg
01-20190511041600.jpg  01-20190511045300.jpg  01-20190511070231.jpg  01-20190511111300.jpg
01-20190511041700.jpg  01-20190511045400.jpg  01-20190511070300.jpg  01-20190511111430.jpg
01-20190511041800.jpg  01-20190511045500.jpg  01-20190511070331.jpg  01-20190511111530.jpg
01-20190511041900.jpg  01-20190511045600.jpg  01-20190511070400.jpg  01-20190511111630.jpg
01-20190511042000.jpg  01-20190511045700.jpg  01-20190511070431.jpg  01-20190511111730.jpg
01-20190511042100.jpg  01-20190511063300.jpg  01-20190511070500.jpg  01-20190511111830.jpg
01-20190511042200.jpg  01-20190511063400.jpg  01-20190511070600.jpg  01-20190511111930.jpg
01-20190511042300.jpg  01-20190511063500.jpg  01-20190511070700.jpg  01-20190511141000.jpg
01-20190511042400.jpg  01-20190511063600.jpg  01-20190511070800.jpg  01-20190511141100.jpg
01-20190511042500.jpg  01-20190511063700.jpg  01-20190511070900.jpg  01-20190608124030.jpg
01-20190511042600.jpg  01-20190511063800.jpg  01-20190511071000.jpg  01-20190608124100.jpg
01-20190511042700.jpg  01-20190511063900.jpg  01-20190511071100.jpg  01-20190608125130.jpg
01-20190511042800.jpg  01-20190511064000.jpg  01-20190511071200.jpg  01-20190608125230.jpg
01-20190511042900.jpg  01-20190511064100.jpg  01-20190511071330.jpg  01-20190608125330.jpg
01-20190511043000.jpg  01-20190511064200.jpg  01-20190511071430.jpg  02-20190608124000.jpg
01-20190511043300.jpg  01-20190511064300.jpg  01-20190511071530.jpg  lastsnap.jpg
01-20190511043400.jpg  01-20190511064400.jpg  01-20190511071630.jpg
01-20190511043500.jpg  01-20190511064500.jpg  01-20190511071730.jpg
01-20190511043600.jpg  01-20190511064600.jpg  01-20190511071830.jpg
```

Рис. 6. Збережені результати розпізнавання

Рис. 7. Порожня мішень

Рис. 8. Мішень із трьома влученнями

2.4. Результати зміни інтерфейсу

Після оновлення зображення, що зберігається під назвою “lastsnap.jpg”, в інтерфейсі користувача відбувається оновлення сторінки на

основі визначених влучень та результатів їх візуалізації на вихідному зображенні (див. рис. 9).

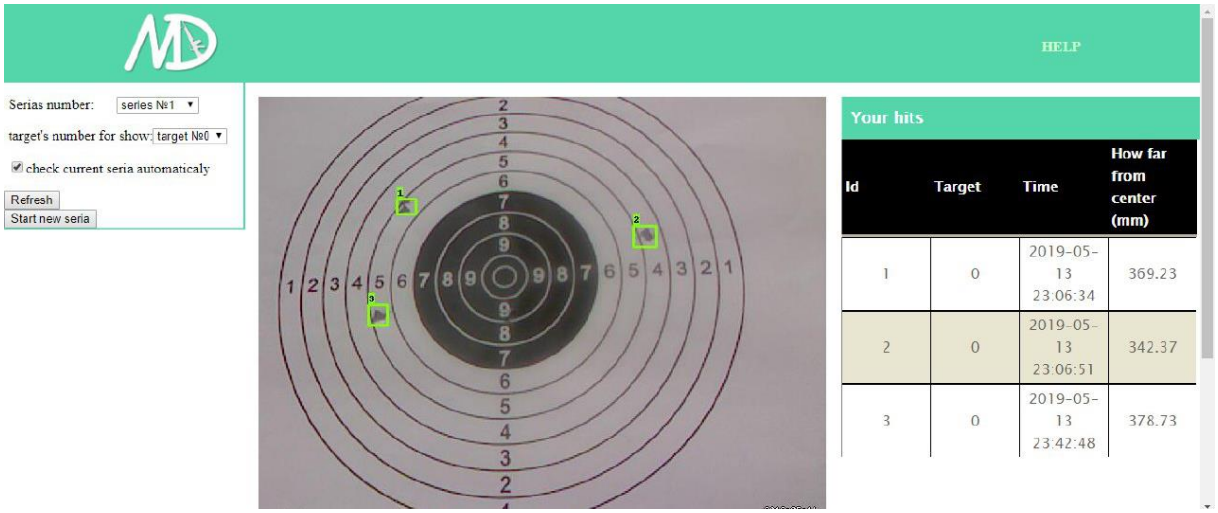


Рис. 9. Результат зміни інтерфейсу